Písomný výstup pedagogického klubu

1. Prioritná os	Vzdelávanie
2. Špecifický cieľ	1.1.1 Zvýšiť inkluzívnosť a rovnaký prístup ku kvalitnému vzdelávaniu a zlepšiť výsledky a kompetencie detí a žiakov
3. Prijímateľ	Stredná priemyselná škola informačných technológií, Nábrežná 1325, Kysucké Nové Mesto
4. Názov projektu	Učme efektívnejšie pre prax
5. Kód projektu ITMS2014+	312011AMJ5
6. Názov pedagogického klubu	Informatika v praxi
7. Meno koordinátora pedagogického klubu	Ing. Peter Remiš
8. Školský polrok	september 2021 – január 2022
9. Odkaz na webové sídlo zverejnenia písomného výstupu	www.spsknm.sk

10.

Úvod:

Stručná anotácia
Výmena vedomostí medzi členmi pedagogického klubu, prehlbovanie vedomostí z oblasti IoT a informatiky, zlepšenie kompetencií a vytvorenie učebného materiálu.

Kľúčové slová
Internet vecí, programovanie, ESP32, Arduino, Raspberry Pi, MIT App Inventor, Linux, IoT, softvér, hardvér, sieť, protokol

Zámer a priblíženie témy písomného výstupu
Zámerom písomného výstupu je tvorba odborných materiálov, ktoré budú slúžiť pre výučbu príbuzných predmetov oblasti IoT. Témy dokumentu sú najmä: sieťové technológie a protokoly, programovanie platformy ESP32, programovanie Android aplikácii pomocou cloudovej služby MIT App Inventor.

Jadro:

Popis témy/problém

Pedagogický klub sa zameral v tomto polroku na oblasti sieťových technológii, programovania ESP32 a tvorby aplikácii pomocou MIT App Inventor. Oblasti boli zvolené na základe výsledkov posledného pedagogického klubu, ktorý sa zaoberal problematikou IoT v širšom kontexte. Tieto jednotlivé oblasti pokrývajú tri zo štyroch vrstiev IoT modelu a síce hardvérovú (zber dát a ich odosielanie pomocou ESP32), sieťovú (prenos pomocou rôznych sieťových technológii) a analytickú (prezentovanie dát a riadenie systému užívateľom Android aplikácie). Príloha k tomuto dokumentu reprezentuje učebnicu, ktorá má slúžiť žiakom počas výučby. Obsahuje nie len teóriu a aktuálne trendy, ale najmä praktické príklady, ktoré môžu žiaci upravovať a tak precvičovať svoje zručnosti v programovaní.

Záver:

Zhrnutia a odporúčania pre činnosť pedagogických zamestnancov

Vďaka pedagogickému klubu si zvýšili učitelia svoje odborné vedomosti zo širokej oblasti. Zároveň sa vytvorili učebné materiály, ktoré slúžia pre výučbu v predmetoch s podobným zameraním, ako je oblasť IoT. Tým sa zmodernizovala výučba, ktorá napomôže k zvýšeniu kvality žiakov a ich uplatneniu v praxi. Obsah študijného materiálu tvorili všetci pedagogický členovia klubu a vychádzali pritom z aktuálneho stavu praxe, no zároveň volili nástroje, ktoré sú pre žiakov oveľa jednoduchšie na výučbu a voľne dostupné. Takýmto príkladom je MIT App Inventor. Ten sa v praxi síce nepoužíva na samotné naprogramovanie Android aplikácie, ale je veľmi vhodným nástrojom pre výučbu programovania inteligentných mobilných aplikácii. Zároveň je Android aplikácia jednou z najžiadanejších foriem, ako prezentovať užívateľovi nejaké dáta a ponúknuť mu možnosť riadenia. Preto voľba nástroja odpovedá aj požiadavke praxe a zároveň aj efektivite výučby žiaka.

11. Vypracoval (meno, priezvisko)	Ing. Peter Remiš
12. Dátum	31.01.2022
13. Podpis	[redacted]
14. Schválil (meno, priezvisko)	Ing. Milan Valek
15. Dátum	31.01.2022
16. Podpis	[redacted]

Stredná priemyselná škola informačných technológií
Nábrežná 1325, 024 01 Kysucké Nové Mesto

INFORMATIKA V PRAXI

Autor: Ing. Peter Remiš

september 2021 – január 2022

OBSAH

1 ESP32

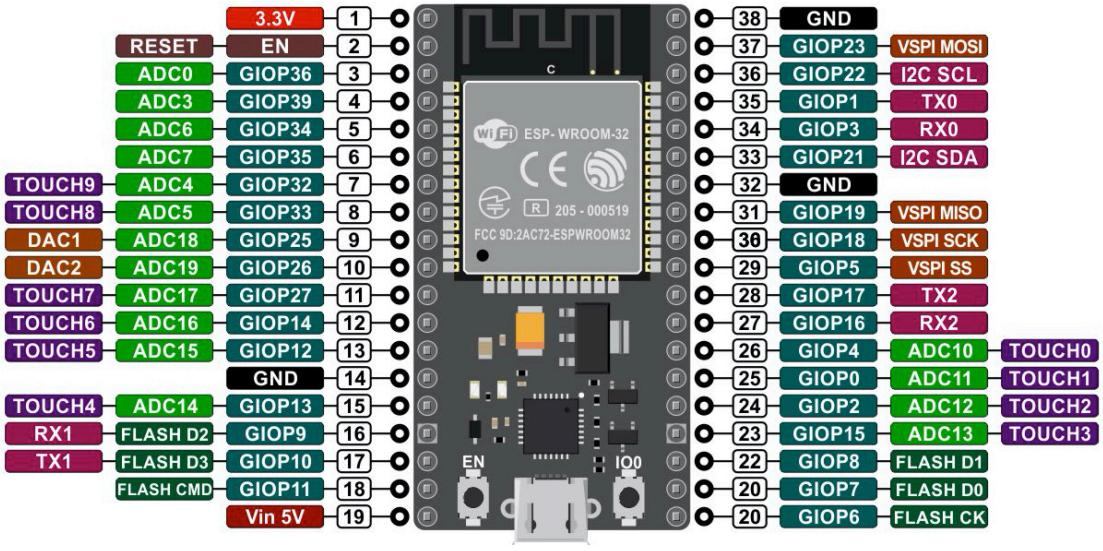
Čip s označením ESP 32 od spoločnosti Espressif Systems podporuje komunikačné rozhrania WiFi aj bluetooth 4.0 LE. Má dvojjadrový procesor s 32 bitovými jadrami Xtensa LX6 taktovanými na 160 MHz. Jedno jadro rieši WiFi komunikáciu a druhé máte k dispozícii pre svoj program. K dispozícii je až 36 GPIO pinov. Kapacita pamäti Flash je 16 MB a RAM je tvorená tromi blokmi s celkovou kapacitou 512 kB.

Porovnanie EPS32 s Arduino UNO:

	Ardunio UNO	**ESP32**
Architektúra	**8 bit**	**32 bit**
RAM	**2 KB**	**512 KB**
Flash	**32 KB**	**16 MB**
CPU frekvencia	**16MHz**	**160MHz**
GPIO PINy	**14**	**36**
Komunikačné zbernice	**SPI, I2C, UART**	**SPI,I2C,UART,I2S,CAN**
Piny AD prevodníka	**6**	**18**
Piny DA prevodníka	**0**	**2**
Logika	**5V/20mA**	**3,3V/20mA**

ESP32 má k dispozícii 34 programovateľných GPIO pinov. Prakticky každý z nich má však aj alternatívnu funkciu, či už v niektorom z komunikačných rozhraní, ako pin pre dotykový senzor a podobne.

.



1.1 Hardvérová výbava ESP 32

- 34 × programovateľných GPIO pinov. Ktorýkoľvek GPIO je možné využiť ako PWM, maximálne však 16 kanálov.
- 12-bit AD prevodník do 18 kanálov. ESP32 disponuje 8 kanálovým ADC1 a 10 kanálovým ADC2 prevodníkom. ADC1 je plne k dispozícii používateľovi. ADC2 prevodník je dostupný obmedzene, pretože väčšinu jeho kanálov využíva modul bezdrôtovej komunikácie.
- 2× 8-bit DA prevodníky
- 10 × dotykové senzory, ktoré dokážu detegovať dotyk prsta na plôške pripojenej k príslušnému pinu
- 4× SPI. Jedno z rozhraní je určené na komunikáciu s flash pamäťou. Pre používateľa teda zostávajú 3 SPI rozhrania.
- 2 × I2S
- 2 × I2C s maximálnou frekvenciou 5 MHz
- 3×UART. Maximálna prenosová rýchlosť je 5 Mbit/s
- CAN 2.0
- Wifi protokoly: 802.11 B/G/N/E/I
- Bluetooth: 2 režimy, tradičný a s nízkou spotrebou
- Integrovaná Hallova sonda. Jej výstup je možné interne pripojiť na zosilňovač a A/D prevodník.
- Je tam však niekoľko obmedzení, ktoré je potrebné poznať:
- Ktorýkoľvek GPIO je možné využiť ako PWM výstup, čiže výstup z pulzno šírkovou moduláciou, maximálne však môžete takto naprogramovať 16 kanálov. Tieto kanály sú číslované 0 – 15.
- Maximálne prúdové zaťaženie GPIO je 40 mA

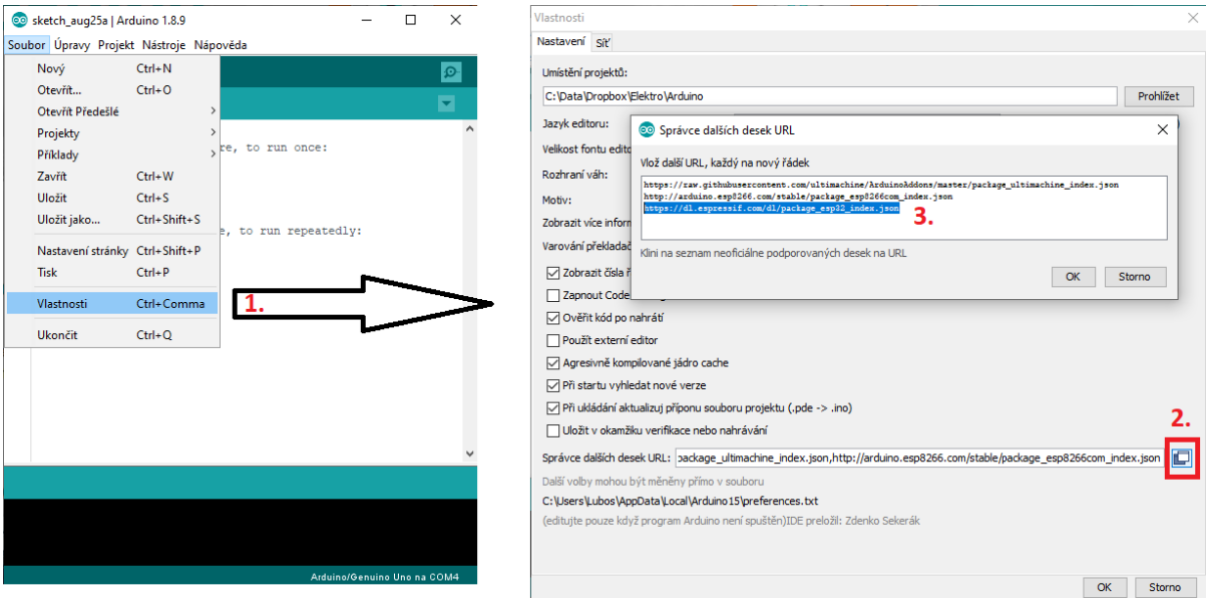
- Porty GPIO 34 - GPIO 39 môžete využiť iba ako vstupné.
- Porty GPIO 6 - GPIO 11 sa využívajú interne na komunikáciu medzi modulom ESP 32 a flash pamäťou
- Na portoch GPIO 1, GPIO 3, GPIO 5, GPIO 14 a GPIO 15 je počas resetu buď logická úroveň HIGH, alebo PWM signál, takže počas bootovania, alebo resetu nemáte zaručenú úroveň na týchto pinoch. Týka sa to aj portov GPIO 6 - GPIO 11, avšak tie nemôžete použiť, viď predchádzajúci bod.
- Piny GPIO 0, GPIO 2 a GPIO 4 sa používajú na nastavenie módu zavádzania firmvéru. Ak k nim máte pripojené periférne zariadenia, môže sa vyskytnúť problém s pokusom o nahranie nového kódu, flashovanie firmvéru alebo resetovaním dosky, pretože aktuálny stav úrovní na pinoch bránia ESP32 vstúpiť do správneho režimu. Po resetovaní, flashovaní, alebo zavedení programu tieto piny fungujú normálne.
- Počas bootovania musia byť piny GPIO 5, GPIO 12 a GPIO 15 v týchto úrovniach: GPIO 5 – HIGH, GPIO 12 – LOW a GPIO 15 – HIGH

1.2 ESP 32 a Arduino IDE

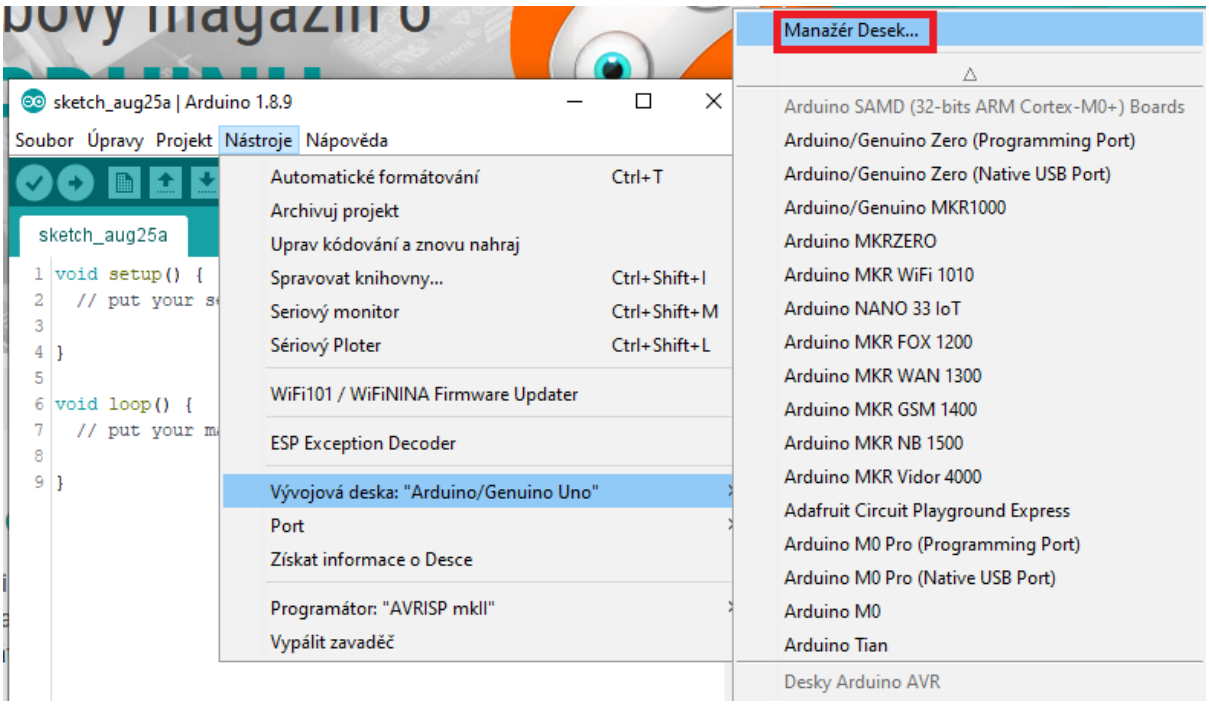
Prvým krokom je pridanie tohto reťazca do Správcu ďalších dosiek:

https://dl.espressif.com/dl/package_esp32_index.json

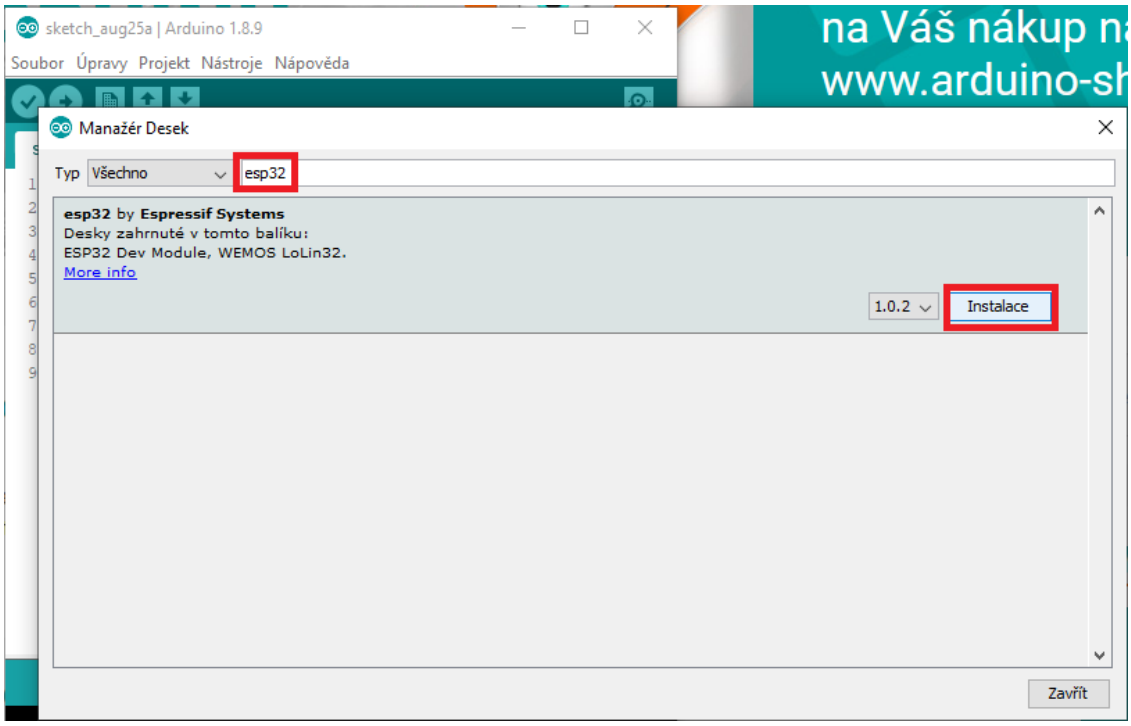
Správca nájdete pri spustení Arduino IDE v ponuke *"Súbor-Vlastnosti"* . Po jeho spustení vložíte spomínaný reťazec do kolónky *"Správca ďalších dosiek URL"* . Pokiaľ už máte pridané iné odkazy, môžete stlačiť ikonku vedľa kolónky a v nej pomocou Enter zadať novú adresu na nový riadok.



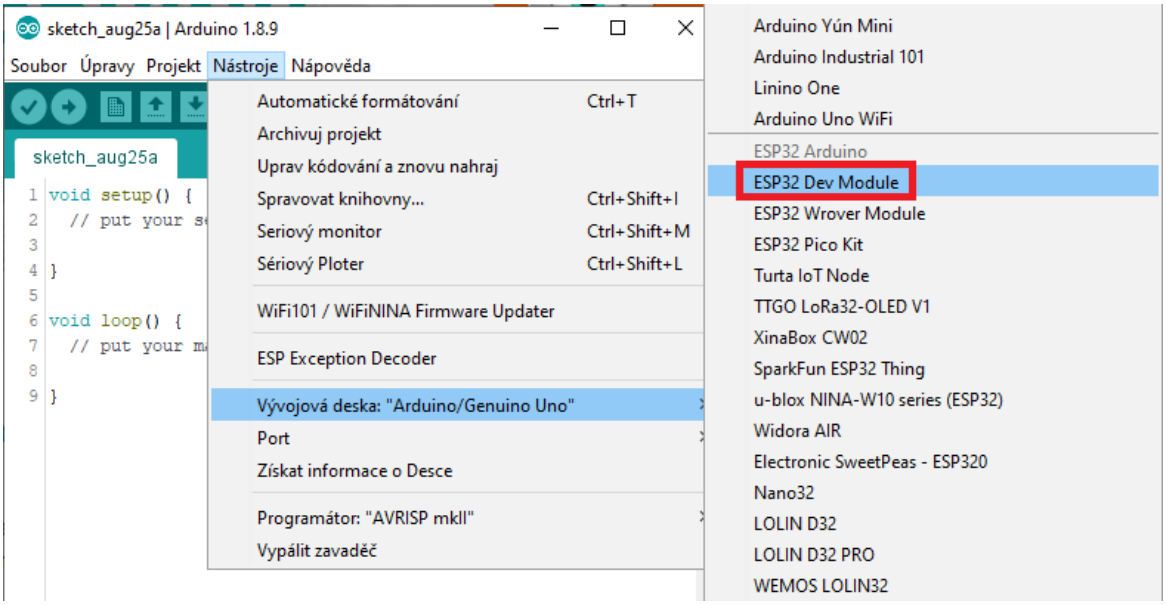
Po zadaní adresy potvrdíme obe ponuky pomocou tlačidla OK a môžeme pridať podporu ESP32 dosiek. To urobíme tak, že otvoríme ponuku *"Nástroje-Vývojová doska-Manažér Dosiek"* .



V novo otvorenom okne s názvom Manažér Dosiek zadáme do vyhľadávacieho riadku názov *"esp32"* a mali by sme vidieť balíček *"esp32 by Espressif Systems"* . U tohto balíka potom stačí na neho prejsť myšou a v pravom dolnej rohu kliknúť na tlačidlo *"Inštalácia"* . Inštalácia môže v závislosti na našej rýchlosti Internete a počítače trvať maximálne niekoľko minút.



Po nainštalovaní už máme hotové a môžeme otestovať podporu tak, že zvolíme nejakú z novo pridaných dosiek. V ponuke *"Nástroje-Vývojová doska"* by sme teda mali vidieť nové dosky *"ESP32 Arduino"* , pričom väčšina predávaných dosiek funguje s univerzálnou voľbou *"ESP32 Dev Module"* .



1.3 Príklad WIFI skener

- Pripojte esp32, system nájde ovládač zariadenia a spustite Ard IDE
- Otvorte súbor esp32_skener WiFI (prekopírovať zo siete do zložky uzivatel/arduino)
- Verifikovať kód a pripadne nakopírovať chýbajúce knižnice
- Počas uploadu do esp 32 stláčať tlačidlo BOOT pri výpise v okne Connecting
- Po ukončení uplodu otvoriť sériový monitor na nadstaviť prenosovú rýchlosť na 115200
- Ak nefunkčné stlačiť reset tlačidlo

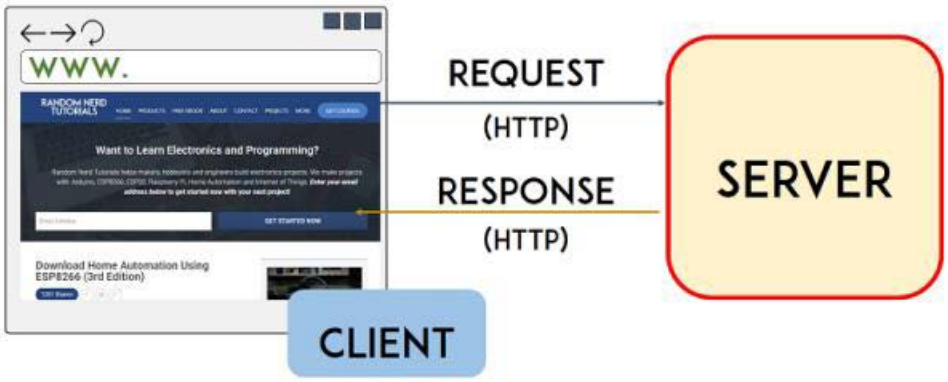
Program

```
#include "WiFi.h"

void setup()
{
   Serial.begin(115200);

   // nastavenie WIFI na STATION MODESet WiFi a odpojenie od AP, ak
predtým bolo ESP32 pripojené k nejakému

   WiFi.mode(WIFI_STA);
   WiFi.disconnect();
   delay(100);

   Serial.println("Setup done");
}

void loop()
{
   Serial.println("scan start");

   // WiFi.scanNetworks vrati pocet najdenych WiFi sieti
   int n = WiFi.scanNetworks();
   Serial.println("scan ukonceny");
```

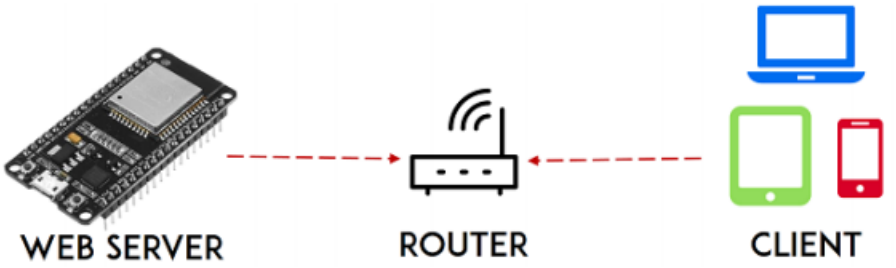
```
if (n == 0)
{
      Serial.println("WiFi siete neboli najdene");
}
else
{
      Serial.print(n);
      Serial.println(" najdenych WiFi sieti");
      for (int i = 0; i < n; ++i)
      {
         // vypis SSID and RSSI pre najdene WiFi siete
         Serial.print(i + 1);
         Serial.print(": ");
         Serial.print(WiFi.SSID(i));
         Serial.print(" (");
         Serial.print(WiFi.RSSI(i));
         Serial.print(")");
         Serial.println(
              (WiFi.encryptionType(i) == WIFI_AUTH_OPEN)?" ":"*");
         delay(50);
      }
   }
   Serial.println("");

   // Počkanie, kým sa znovu spustí skener
   delay(5000);
}
```

1.4 Príklad WEB server

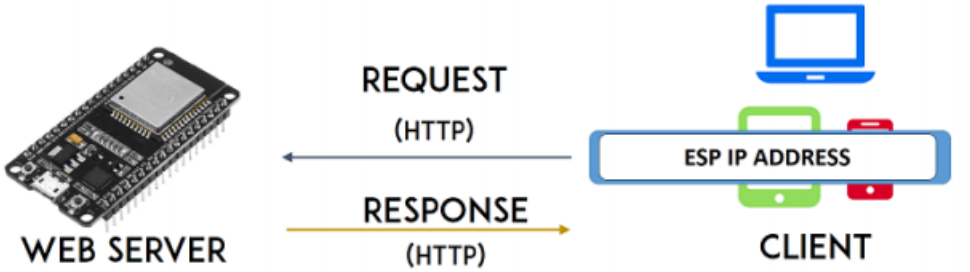
Ak zadáte URL do webového prehliadača, posiela client požiadavku prostredníctvom HTTP na server. Keď server prijme požiadavku, odpovedá prehliadaču zase prostredníctvom HTTP. Client a server komunikujú v počítačovej sieti.



ESP32 vytvára v lokálnej sieti web server, ktorý je pripojený prostredníctvom WiFi na router. V tej istej sieti sa nachádza iné zariadenie s webovým prehliadačom.



Po zadaní IP adresy do prehliadača, tento pošle požiadavku prostredníctvom HTTP do ESP32. Po prijatí ESP 32 odosiela odpoveď v podobe HTTP do webového prehliadača v podobe webovej stránky s odpovedajúcim obsahom.



Program pre HTML prehliadač

```
<!DOCTYPE html> //hovori web prehliadacu, ze ide o html subor
<html> //indikuje zaciatok web stranky
<head> //hlavicka web stranky - zaciatok
<title>ESP32 Web Server</title> // titulok vo webovom prehlaidaci
<meta name="viewport" content="width=device-width, initial-scale=1">
//metadata urobia respozivnost stranky
<link rel="icon" href="data:,">
//favicon je ikona odkazu, ktorá sa zobrazuje na karte webového prehliadača
<style> // definovanie kaskadoveho stylu CSS
html { // styl htlm ma nasledujuce parametre
font-family: Helvetica;
display: inline-block;
margin: 0px auto;
text-align: center;
}
.button { // styl .button ma nasledujuce parametre
background-color: #4CAF50;
```

```
border: none;
color: white;
padding: 16px 40px;
text-decoration: none;
font-size: 30px;
margin: 2px;
cursor: pointer;
}
.button2 { // styl .button2 ma nasledujuce parametre
background-color: #555555;
}
</style>
</head> //hlavicka web stranky - koniec

<body> //telo web stranky - zaciatok
<h1>ESP32 Web Server</h1> //headings - nadpisy
<p>GPIO 26 - State</p> //paragraphs - odseky
<p><a href="/26/on"><button class="button">ON</button></a></p>
//buttons - tlacidla
<p><a href="/26/off"><button class="button button2">OFF</button></a></p>
//<a href= volanie hyperlinku
<p>GPIO 27 - State</p>
<p><a href="/27/on"><button class="button">ON</button></a></p>
//class volanie definovaneho stylu v CSS
<p><a href="/27/off"><button class="button button2">OFF</button></a></p>
</body> //telo web stranky - koniec
</html> //indikuje koniec web stranky
```

Popis programu:

Načítanie WiFI knižnice.

#include <WiFi.h>

Do nasledujúcich riadkov je potrebné vložiť vaše SSID a heslo vo vnútri úvodzoviek.

const char ssid = "";*

const char password = "";*

Potom nastavíte webový server na port 80.

WiFiServer server(80);

Nasledujúci riadok vytvára premennú na uloženie hlavičky požiadavky HTTP:

String header;

Ďalej vytvoríte pomocné premenné na uloženie aktuálneho stavu vašich výstupov.

String output26State = "off";

String output27State = "off";

Každému z vašich výstupov tiež musíte priradiť GPIO.

const int output26 = 26;

const int output27 = 27;

setup ()

Najskôr začneme sériovú komunikáciu s prenosovou rýchlosťou 115 200 na účely ladenia.

Serial.begin(115200);

Taktiež definujete svoje GPIO ako VÝSTUPY a nastavíte ich na LOW.

pinMode(output26, OUTPUT);

pinMode(output27, OUTPUT);

digitalWrite(output26, LOW);

digitalWrite(output27, LOW);

Nasledujúce riadky začínajú Wi-Fi pripojenie s WiFi.begin (ssid, heslo), počkajú na úspešné pripojenie a vypíšu IP adresu ESP do Serial Monitora.

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {

delay(500);

Serial.print(".");

}

// Print local IP address and start web server

Serial.println("");

Serial.println("WiFi connected.");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

server.begin();

loop ()

Definujeme, čo sa stane, keď nový klient nadviaže spojenie s webový serverom. ESP neustále zisťuje prichádzajúcich klientov pomocou tohto riadku:

WiFiClient client = server.available();

Keď prijmeme požiadavku od klienta, uložíme prichádzajúce údaje. Slučka while je aktívna, pokiaľ bude klient pripojený. Neodporúčame zmenu nasledujúcej časti kódu!

if (client) { // If a new client connects,

Serial.println("New Client."); // print a message out in the serial port

String currentLine = ""; // make a String to hold incoming data

while (client.connected()) { // loop while the client's connected

if (client.available()) { // if there's bytes to read from client

char c = client.read(); // read a byte, then

```
Serial.write(c); // print it out the serial monitor

header += c;

if (c == '\n') { // if the byte is a newline character

// if line is blank, you got two newline characters in a row.

// that's the end of the client HTTP request, so send a response:

if (currentLine.length() == 0) {

// HTTP headers start with a response code (e.g. HTTP/1.1 200 OK)

// and a content-type so the client knows what's coming

client.println("HTTP/1.1 200 OK");

client.println("Content-type:text/html");

client.println("Connection: close");

client.println();
```

Ďalšia časť príkazov if and else kontroluje, ktoré tlačidlo bolo stlačené vo vašej webovej stránke a podľa toho riadi výstupy. Ako bolo možné vidieť, vytvárame žiadosť o rôzne adresy URL v závislosti od tlačidla, ktoré stlačíme.

```
// turns the GPIOs on and off

if (header.indexOf("GET /26/on") >= 0) {

Serial.println("GPIO 26 on");

output26State = "on";

digitalWrite(output26, HIGH);

} else if (header.indexOf("GET /26/off") >= 0) {

Serial.println("GPIO 26 off");
```

```
output26State = "off";

digitalWrite(output26, LOW);

} else if (header.indexOf("GET /27/on") >= 0) {

Serial.println("GPIO 27 on");

output27State = "on";

digitalWrite(output27, HIGH);

} else if (header.indexOf("GET /27/off") >= 0) {

Serial.println("GPIO 27 off");

output27State = "off";

digitalWrite(output27, LOW);

}
```

Napríklad, ak ste stlačili tlačidlo GPIO 26 ON, ESP dostane požiadavku na / 26 / ON URL a tieto informácie dostaneme do hlavičky HTTP. Môžeme teda skontrolovať, či hlavička obsahuje výraz GET / 26 / on. Ak obsahuje, vytlačí správu na serveri Sériový monitor, zmení výstupnú premennú 26 na ZAPNUTÉ a rozsvieti sa LED dióda. Toto funguje podobne aj pre ostatné tlačidlá. Ak teda chcete pridať ďalšie výstupy, mali by ste upravte túto časť kódu tak, aby ich obsahovala.

Zobrazenie webovej stránky HTML

Ďalšia vec, ktorú musíte urobiť, je vytvorenie webovej stránky. ESP32 bude posielať a odpovedať na váš prehliadač pomocou kódu HTML na vytvorenie webovej stránky. Webová stránka sa odošle klientovi pomocou výrazu client.println (). Ako argument zadajte to, čo chcete klientovi poslať.

Prvá vec, ktorú by sme mali poslať, je vždy nasledujúci riadok, ktorý naznačuje, že sme odosielanie HTML.

```
<!DOCTYPE HTML><html>
```

Nasledujúci riadok potom zabezpečí, aby sa webová stránka zobrazovala v ľubovoľnom webovom prehliadači.

client.println("<head><meta name=\"viewport\" content=\"width=devicewidth,

initial-scale=1\">");

A nasledujúce sa používa na zabránenie požiadavkám na favicon. Je to ikona odkazu, ktorá sa zobrazuje na karte webového prehliadača.

client.println("<link rel=\"icon\" href=\"data:,\">");

Styling webovej stránky

Ďalej máme text CSS, ktorý upravuje štýl tlačidiel a vzhľadu webovej stránky. My zvolíme písmo Helvetica, definujeme obsah, ktorý sa má zobraziť ako blok a zarovnanie -centrovanie.

client.println("<style>html { font-family: Helvetica; display:

inline-block; margin: 0px auto; text-align: center;}");

Naše tlačidlá štylizujeme farbou # 4CAF50, bez orámovania, textom v bielej farbe a znakom toto polstrovanie: 16px 40px. Nastavili sme tiež bordura - žiadna, určili sme veľkosť písma, znak okraja a kurzor na ukazovateľ.

client.println(".button { background-color: #4CAF50; border: none; color:

white; padding: 16px 40px;");

client.println("text-decoration: none; font-size: 30px; margin: 2px;

cursor: pointer;}");

Definujeme tiež štýl druhého tlačidla so všetkými jeho vlastnosťami ako predtým, ale s inou farbou. Toto bude štýl pre tlačidlo vypnutia.

client.println(".button2{background-color:#555555;}</style></head>");

V ďalšom riadku môžete nastaviť prvý nadpis svojej webovej stránky. Máme tu „ESP32 Web Server “, ale tento text môžete zmeniť podľa potreby.

```
// Web Page Heading

client.println("<body><h1>ESP32 Web Server</h1>");
```

Potom napíšete odsek, ktorý zobrazí aktuálny stav GPIO 26. Ako vidíte, používame output26Statevariable, aby sa stav okamžite aktualizoval, keď sa táto premenná zmení.

```
client.println("<p>GPIO 26 - State " + output26State + "</p>");
```

Potom zobrazíme tlačidlo zapnutia alebo vypnutia v závislosti od aktuálneho stavu GPIO. Ak aktuálny stav GPIO je vypnutý, zobrazíme tlačidlo ON, ak nie, zobrazíme OFF tlačidlo.

```
if (output27State=="off") {

client.println("<p><a href=\"/27/on\"><button

class=\"button\">ON</button></a></p>");

} else {

client.println("<p><a href=\"/27/off\"><button class=\"button

button2\">OFF</button></a></p>");

}
```

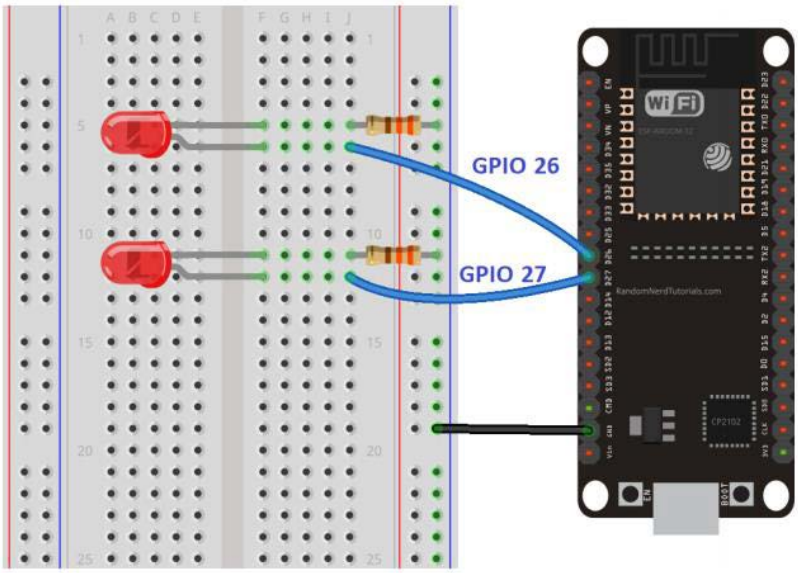
Rovnaký postup používame aj pri GPIO 27.

Nakoniec, keď odpoveď skončí, vyčistíme premennú hlavičky a zastavíme spojenie

s klientom s client.stop ().

```
header = ""; // Clear the header variable

client.stop(); // Close the connection
```

Schéma



ESP 32 web server - 2 tlačidlá

```
#include <WiFi.h>

const char* ssid = "xxxxxxx";

const char* password = "xxxxxxxx";

// nastavenie web server na port port 80

WiFiServer server(80);

// Premenná na uloženie hlavicky požiadavky HTTP

String header;

// Pomocné premenné na uloženie aktuálneho stavu výstupu

String output26State = "off";

String output27State = "off";

// Pomocné premenné na uloženie pinov výstupu GPIO

const int output26 = 25;
```

```
const int output27 = 33;

// Aktuálny čas

unsigned long currentTime = millis();

// Predchádzajúci čas

unsigned long previousTime = 0;

// Definujte časový limit v milisekundách (example: 2000ms = 2s)

const long timeoutTime = 2000;

void setup() {

Serial.begin(115200);

// Inicializujte výstupné premenné ako výstupy

pinMode(output26, OUTPUT);

pinMode(output27, OUTPUT);

// Nastavenie vystupov na log 0

digitalWrite(output26, LOW);

digitalWrite(output27, LOW);

// Pripojenie sa k sieti Wi-Fi pomocou identifikátora SSID a hesla

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {

delay(500);
```

```
Serial.print(".");

}

// Vypisanie lokalnej IP adresy a spustenie weboveho servera

Serial.println("");

Serial.println("WiFi connected.");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

server.begin();

}

void loop(){

WiFiClient client = server.available(); // Sledovanie prichádzajúcich klientov

if (client) { // Ak sa pripojí nový klient,

currentTime = millis();

previousTime = currentTime;

Serial.println("New Client."); //vypíšeme správu na sériovom porte

String currentLine = ""; // vytvoríme String na uchovanie //prichádzajúcich dát od klienta

while (client.connected() && currentTime - previousTime <= timeoutTime) { // slučka, kým je klient //pripojený

currentTime = millis();

if (client.available()) { // ak existujú bajty na čítanie od klienta,

char c = client.read(); // potom prečítaj bajt a
```

```
Serial.write(c); // vypis na sériovom monitore

header += c;

if (c == '\n') { // ak je bajt znakom nového riadku

//ak je aktuálny riadok prázdny, nasleduju za sebou dva znaky nového riadku a

// to je koniec požiadavky od HTTP klienta, preto sa posle odpoveď:

if (currentLine.length() == 0) {

// Hlavičky HTTP vždy začínajú kódom odpovede (napr. HTTP / 1,1 200 OK)

// a typom obsahu, aby klient vedel, čo príde. potom nasleduje prázdny riadok:

client.println("HTTP/1.1 200 OK");

client.println("Content-type:text/html");

client.println("Connection: close");

client.println();

// zapína a vypína GPIO a vytvárame žiadosť o rôzne adresy URL v závislosti od stlac tlačidla

if (header.indexOf("GET /26/on") >= 0) {

Serial.println("GPIO 26 on");

output26State = "on";

digitalWrite(output26, HIGH);

} else if (header.indexOf("GET /26/off") >= 0) {

Serial.println("GPIO 26 off");

output26State = "off";

digitalWrite(output26, LOW);
```

```
} else if (header.indexOf("GET /27/on") >= 0) {

Serial.println("GPIO 27 on");

output27State = "on";

digitalWrite(output27, HIGH);

} else if (header.indexOf("GET /27/off") >= 0) {

Serial.println("GPIO 27 off");

output27State = "off";

digitalWrite(output27, LOW);

}

// Zobrazenie webovej stránky HTML

client.println("<!DOCTYPE html><html>");

client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");

client.println("<link rel=\"icon\" href=\"data:,\">");

- State " + output26State + "</p>");

// Ak je výstup 26State vypnutý, zobrazuje tlačidlo ON

if (output26State=="off") {

client.println("<p><a href=\"/26/on\"><button class=\"button\"> ON </button></a></p>");

} else {

client.println("<p><a href=\"/26/off\"><button class=\"button button2\"> OFF </button></a></p>");

}
```

```
// Zobrazenie aktuálneho stavu a tlačidlá ON / OFF pre GPIO 27

client.println("<p>GPIO 27 - State " + output27State + "</p>");

// Ak je výstup27State vypnutý, zobrazí sa tlačidlo ZAP

if (output27State=="off") {

client.println("<p><a href=\"/27/on\"><button class=\"button\"> ON </button></a></p>");

} else {

client.println("<p><a href=\"/27/off\"><button class=\"button button2\"> OFF </button></a></p>");

}

client.println("</body></html>");

// Odpoveď HTTP sa končí ďalším prázdnym riadkom

client.println();

// Ukoncenie slučky while

break;

} else { // ak bol prijaty nový riadok, potom vymažte currentLine

currentLine = "";

}

} else if (c != '\r') { // ak bol prijaty iny znak, ako znak návratu, znak sa prida

currentLine += c; // sa prida na koniec currentLine

}

}
```

```
}

// Vymazanie premennej hlavičky

header = "";

// Ukoncenie pripojenie

client.stop();

Serial.println("Client disconnected.");

Serial.println("");

}
```

1.5 Príklad merania fyzikálnych veličín pomocou DHT11

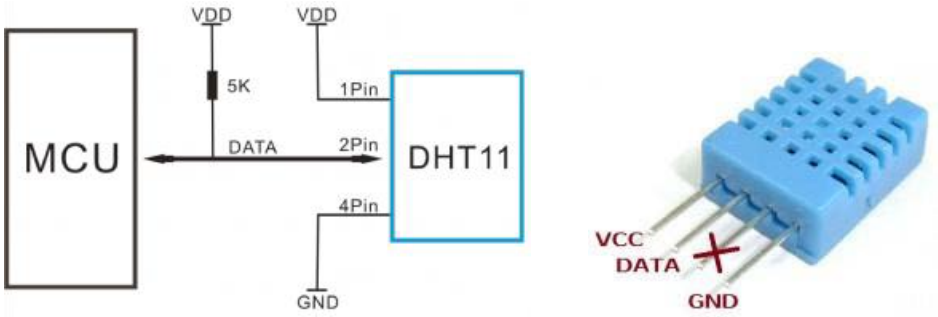
Nasledujúci príklad vychádza z predošlého, preto netreba tak podrobný popis. Rozbor kódu je písaný ako komentár v kóde.

Parametre DHT11

- Merací rozsah: 20-90% RH; 0-50 °C
- Presnosť merania vlhkosti: ±5% RH
- Presnosť merania teploty: ±2 %
- Napájacie napätie: 3,3 – 6V

Piny modulu DHT11

- DATA - pripája sa na pin G17
- GND - elektrická zem
- VCC - Napájacie napätie 3,3 V



Program

#include <WiFi.h> // nacitanie kniznice pre Wi-Fi

#include <Wire.h>

#include <dht11.h> // importuje knižnicu DHT11

dht11 MojSenzor; //vytvorí objekt DHT11 s názvom MojSenzor

const char ssid = "xxxxxxxx";*

const char password = "xxxxxxxx";*

WiFiServer server(80);

String header;

unsigned long currentTime = millis();

unsigned long previousTime = 0;

const long timeoutTime = 2000;

void setup() {

Serial.begin(115200);

bool status;

Serial.print("Connecting to ");

```
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
// Print local IP address and start web server
Serial.println("");
Serial.println("WiFi connected.");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
server.begin();
}
void loop(){
MojSenzor.read(33); // prečíta údaje zo senzoru DTH11 pripojeneho na pin G33
int teplota = MojSenzor.temperature; // načítanie teploty
int vlhkost = MojSenzor.humidity; // načítanie vlhkosti
WiFiClient client = server.available(); // Listen for incoming clients
if (client) { // If a new client connects,
currentTime = millis();
previousTime = currentTime;
```

Serial.println("New Client."); // print a message out in the serial port

String currentLine = ""; // make a String to hold incoming data from the client

while (client.connected() && currentTime - previousTime <= timeoutTime) { // loop while the client's connected

currentTime = millis();

if (client.available()) { // if there's bytes to read from the client,

char c = client.read(); // read a byte, then

Serial.write(c); // print it out the serial monitor

header += c;

if (c == '\n') { // if the byte is a newline character

// if the current line is blank, you got two newline characters in a row.

// that's the end of the client HTTP request, so send a response:

if (currentLine.length() == 0) {

// HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)

// and a content-type so the client knows what's coming, then a blank line:

client.println("HTTP/1.1 200 OK");

client.println("Content-type:text/html");

client.println("Connection: close");

client.println();

// Display the HTML web page

client.println("<!DOCTYPE html><html>");

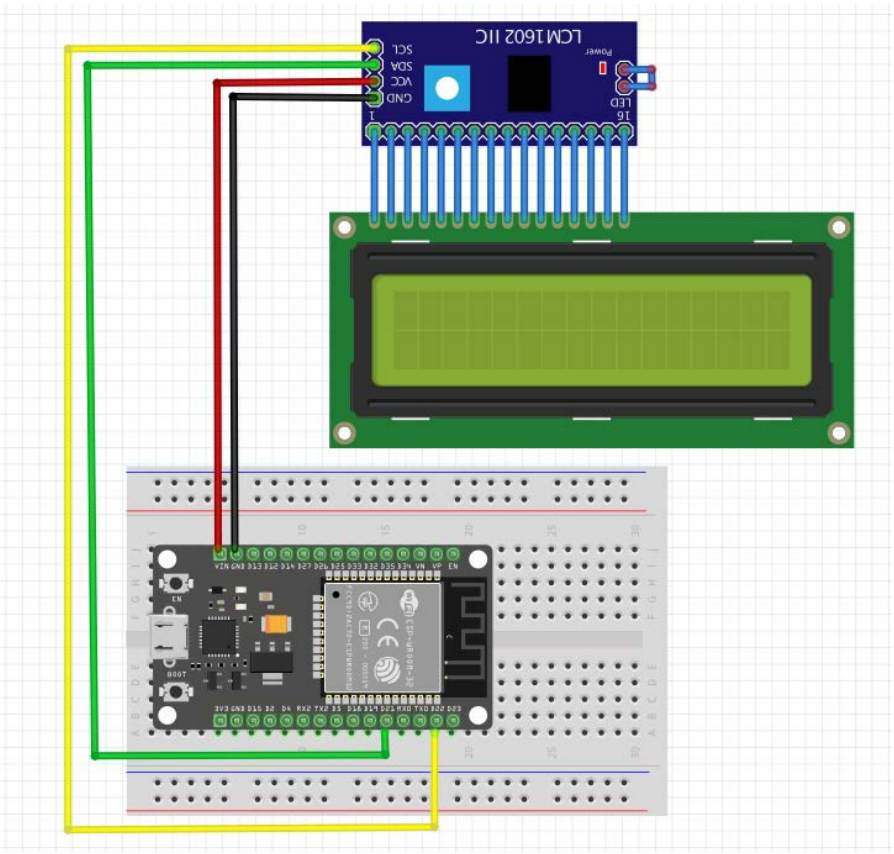
```
client.println("<head><meta  name=\"viewport\" content=\"width=device-width, initial- scale=1\">");

client.println("<link rel=\"icon\" href=\"data:,\">");

// CSS to style the table

client.println("<style>body { text-align: center; font-family: \"Trebuchet MS\", Arial;}");

client.println("table { border-collapse: collapse; width:35%; margin-left:auto; margin-right:auto; }");

client.println("th { padding: 12px; background-color: #0043af; color: white; }");

client.println("tr { border: 1px solid #ddd; padding: 12px; }");

client.println("tr:hover { background-color: #bcbcbc; }");

client.println("td { border: none; padding: 12px; }");

client.println(".sensor { color:white; font-weight: bold; background-color: #bcbcbc; padding: 1px; }");

// Web Page Heading

client.println("</style></head><body><h1>ESP32 with DHT11</h1>");

client.println("<table><tr><th>MEASUREMENT</th><th>VALUE</th></tr>");

client.println("<tr><td>Temp. Celsius</td><td><span class=\"sensor\">");

client.println(teplota);

client.println(" *C</span></td></tr>");

client.println("<tr><td>Temp. Fahrenheit</td><td><span class=\"sensor\">");

client.println(1.8 * teplota + 32);
```

```
client.println(" *F</span></td></tr>");

client.println("<tr><td>Humidity</td><td><span class=\"sensor\">");

client.println(vlhkost);

client.println(" %</span></td></tr>");

client.println("</body></html>");

// The HTTP response ends with another blank line

client.println();

// Break out of the while loop

break;

} else { // if you got a newline, then clear currentLine

currentLine = "";

} } else if (c != '\r') { // if you got anything else but a carriage return character,

currentLine += c; // add it to the end of the currentLine

} } }

// Clear the header variable

header = "";

// Close the connection

client.stop();

Serial.println("Client disconnected.");

Serial.println("");

} }
```

1.6 ESP 32 a LCD displej

LCD displeje Rozdeľujú sa na grafické a znakové. Grafický displej umožňuje na kontinuálnej matici bodov zobraziť grafické obrazce a samozrejme aj text rôznym fontom. Znakové displeje sú v podstate tiež maticové, akurát matice pre každý znak sú oddelené. Znakové displeje umožňujú zobraziť rôzne dlhý text v jednom, alebo viacerých riadkoch. Displeje sú k dispozícii v konfigurácii 8x1, čiže 8 znakov v jednom riadku až po 40 x 4, čiže 4 riadky, pričom každý z nich môže mať 40 znakov. Programové ovládanie týchto displejov je pomerne jednoduché. Nastavíte kurzor na miesto kde chcete mať znak, alebo začiatok textu a pošlete znak, alebo text.

Alfanumerické displeje využívajú ako radič obvod Hitachi HD44780. Radič má riadiaci a dátový register. V pamäti ROM má napevno uložené definície 208 znakov v matici 5 × 8 bodov a 32 znakov v matici 5 × 10 bodov. Zlou správou pre našinca je, že tam nie sú znaky so slovenskou diakritikou. Tie môžete v matici 5 × 8 bodov definovať do pamäte typu CGRAM. Obrazová pamäť RAM obsahuje aktuálne zobrazený text. Displej tohoto typu môžete pripojiť aj cez sériovú zbernicu I2C pomocou dvoch dátových pinov (SDA a SCL) + dvoch pinov na napájanie.



Displej sa k mikrokontroléru pripája pomocou štyroch vodičov a využíva rozhranie I2C. Displej vyžaduje napájacie napätie 5V. Display pracuje s 5V logikou, ale SCL a SDA sú vstupy, preto nie je pozrebný prevodník úrovní (resp delič napätia).

- SCL - pripája sa na pin G22, ktorý má alternatívnu funkciu I2C_SCL
- SDA - pripája sa na pin G21, ktorý má alternatívnu funkciu I2C_SDA
- GND - elektrická zem
- VCC - Napájacie napätie 5 V

Pri uploade musí byť odpojené napájanie displeja od svorky Vin5! Zaťažovaný stabilizátor napätia v esp32. Po pripojení napájania je nutne resetovať ESP 32!

Príklad vypísania teploty na LCD

```
#include <Wire.h> // pridaj knižnicu I2C
#include <LiquidCrystal_I2C.h> // pridaj knižnicu I2C dispeja
#include <dht11.h> // importuje knižnicu DHT11
dht11 MojSenzor; //vytvori objekt DHT11 s názvom MojSenzor
LiquidCrystal_I2C lcd(0x27,16,2); //nastavenie adresy na 0x27 pre 16x2 disp.


void setup()
{
lcd.init(); // inicializacia lcd
lcd.backlight(); // zapnúť podsvietenie
}
```

```
void loop()
{
MojSenzor.read(17); // prečíta údaje zo senzoru DTH11 pripojeneho na pin G17
int teplota = MojSenzor.temperature; // načítanie teploty
lcd.setCursor(3,0); // nastaví kurzor na stlpec 4 riadok 1
lcd.print("t = "); // napíše text t =
lcd.print(teplota); // napíše hodnotu teploty
lcd.print(" oC"); // napíše oC
delay(300); // počkať 300 ms
}
```

Príklad vypísania teploty a vlhkosti na LCD

```
#include <Wire.h> // pridaj knižnicu I2C
#include <LiquidCrystal_I2C.h> // pridaj knižnicu I2C dispeja
#include <dht11.h> // importuje knižnicu DHT11
dht11 MojSenzor; //vytvori objekt DHT11 s názvom MojSenzor
LiquidCrystal_I2C lcd(0x27,16,2); // nastavenie adresy na 0x27 pre 1602 displej
void setup() {
lcd.init(); // inicializacia lcd
lcd.backlight(); // zapnúť podsvietenie
}
```

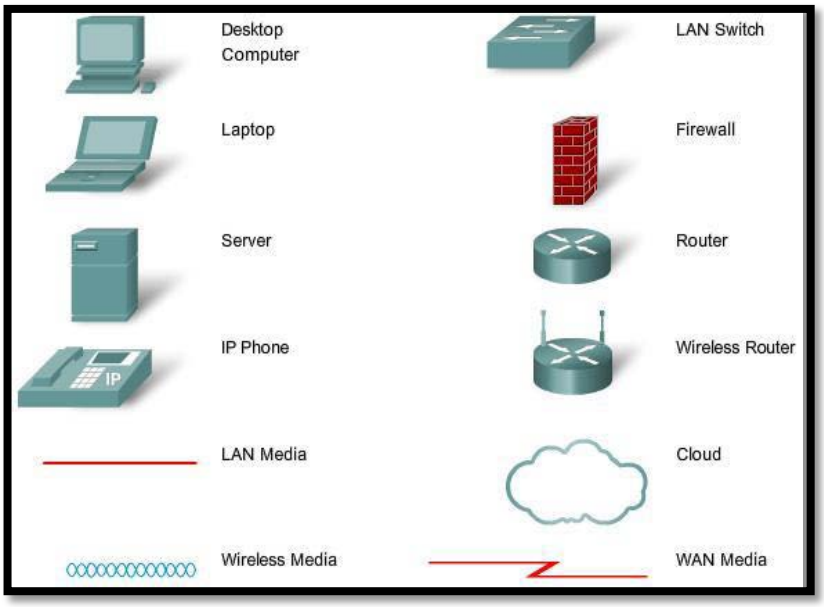
```
void loop()

{

MojSenzor.read(17); // prečíta údaje zo senzoru DTH11 pripojeneho na pin G17

int teplota = MojSenzor.temperature; // načítanie teploty

int vlhkost = MojSenzor.humidity; // načítanie vlhkosti

lcd.setCursor(3,0); // nastaví kurzor na stlpec 4 riadok 1

lcd.print("t = "); // napíše text t =

lcd.print(teplota); // napíše hodnotu teploty

lcd.print(" oC"); // napíše oC

lcd.setCursor(3,1); // nastaví kurzor na stlpec 4 riadok 2

lcd.print("v = "); // napíše text v =

lcd.print(vlhkost); // napíše hodnotu teploty

lcd.print(" % "); // napíše %

delay(300); // počkať 300 ms

}
```

2 POČÍTAČOVÉ SIETE

Počítačová sieť vzniká prepojením dvoch zariadení, ktoré si medzi sebou vymieňajú informácie.

Každá počítačová sieť obsahuje tieto komponenty:

- **Koncové zariadenia**
- **Medziľahlé zariadenia**
- **Sieťové média**

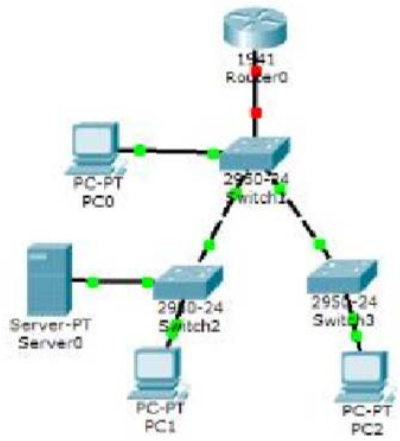


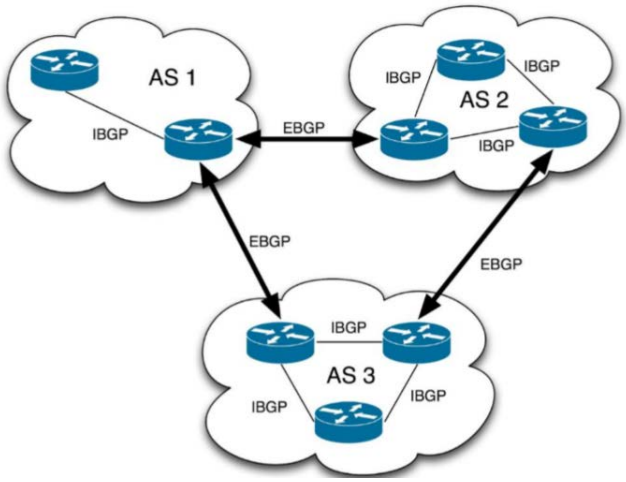
Najznámejšie typy sieťovej infraštruktúry

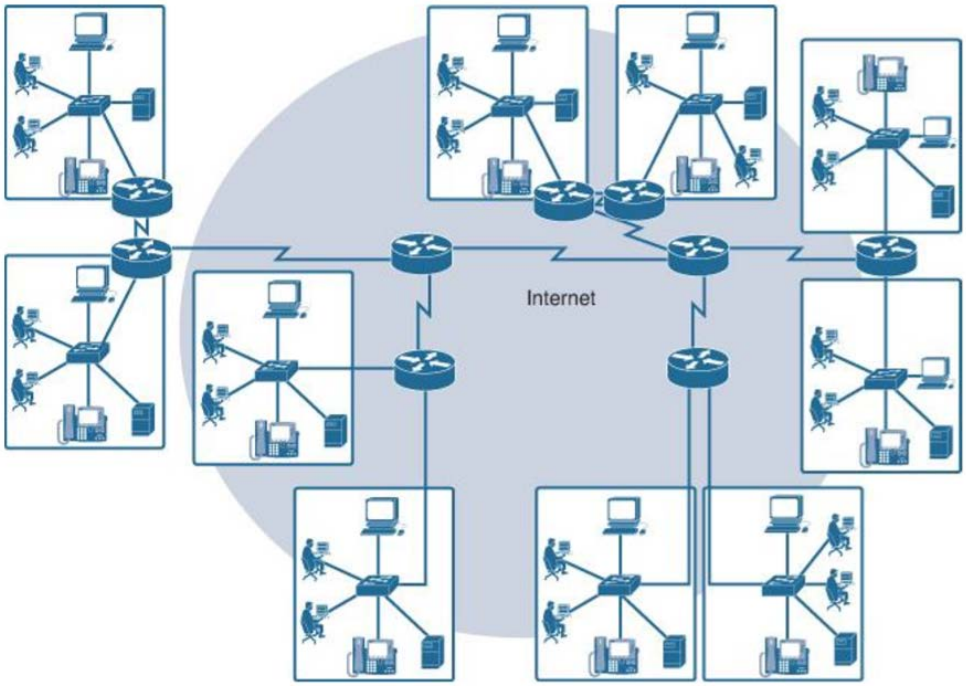
- LAN (Local Area Network)
- WAN (Wide Area Network)
- Internet

Intranet: je akákoľvek vnútorná firemná sieť. Je súkromné pripojenie LAN a WAN, ktoré patria do organizácie a je navrhnuté tak, aby bolo prístupné len pre členov organizácie, zamestnancov alebo pre osoby s povolením

Extranet: je sieť inej firmy, ktorý umožňuje obmedzený a kontrolovaný prístup do nášho vlastného internetu.



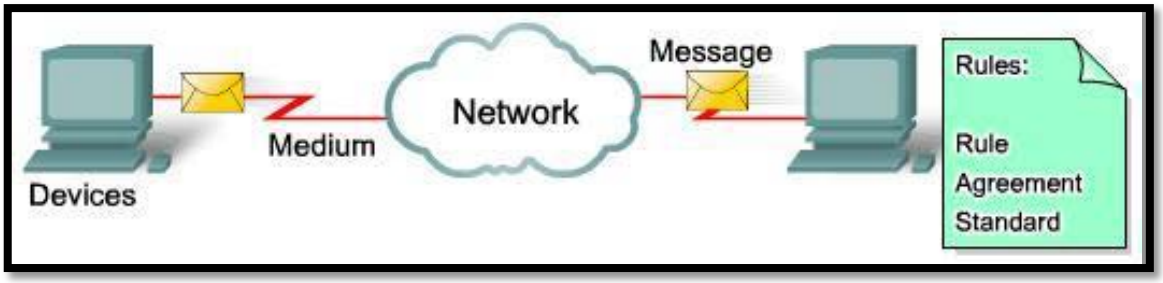




Sieťová komunikácia

Odosielanie správy, či už prostredníctvom osobnej komunikácie alebo prostredníctvom siete, sa riadi pravidlami, ktoré sa nazývajú protokoly. Tieto protokoly sú špecifické pre typ použitej komunikačnej metódy.

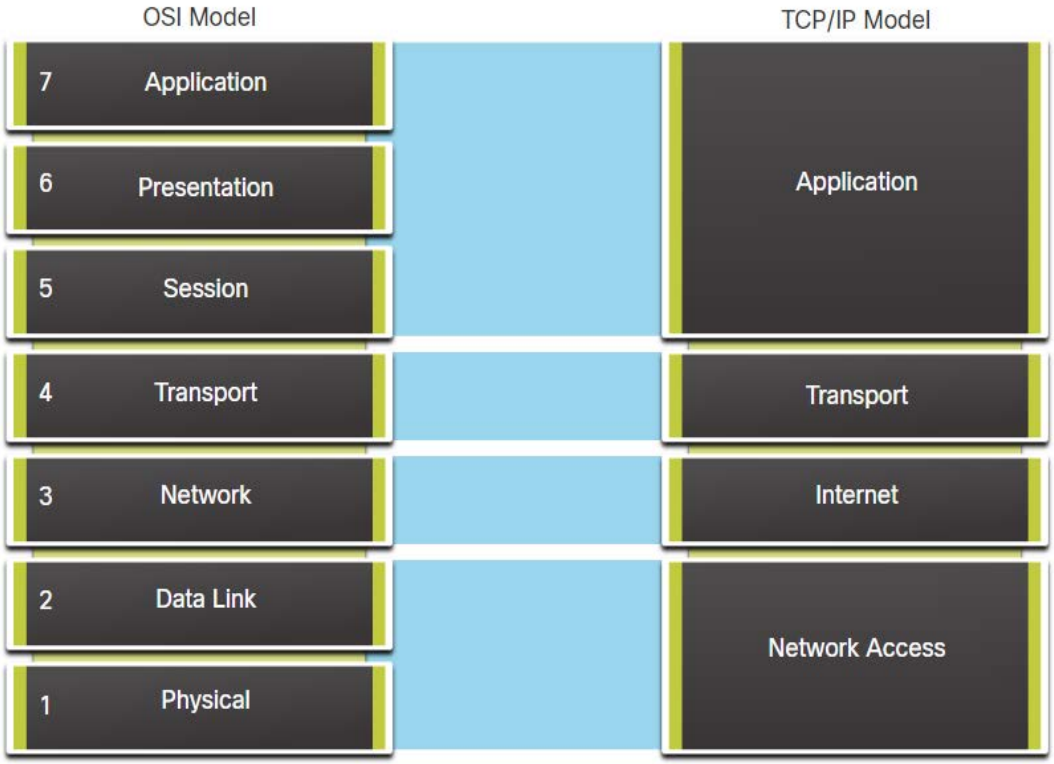
Úspešná komunikácia sa musí riadiť istými pravidlami. V počítačových sieťach tieto pravidla nazývame protokoly (Protocol) alebo sieťové protokoly (Network Protocol). Sieťové protokoly definujú spoločný formát a syntax prenášanej správy, význam správy, spôsob prenosu, spôsob spracovania obsahu prenášanej správy.

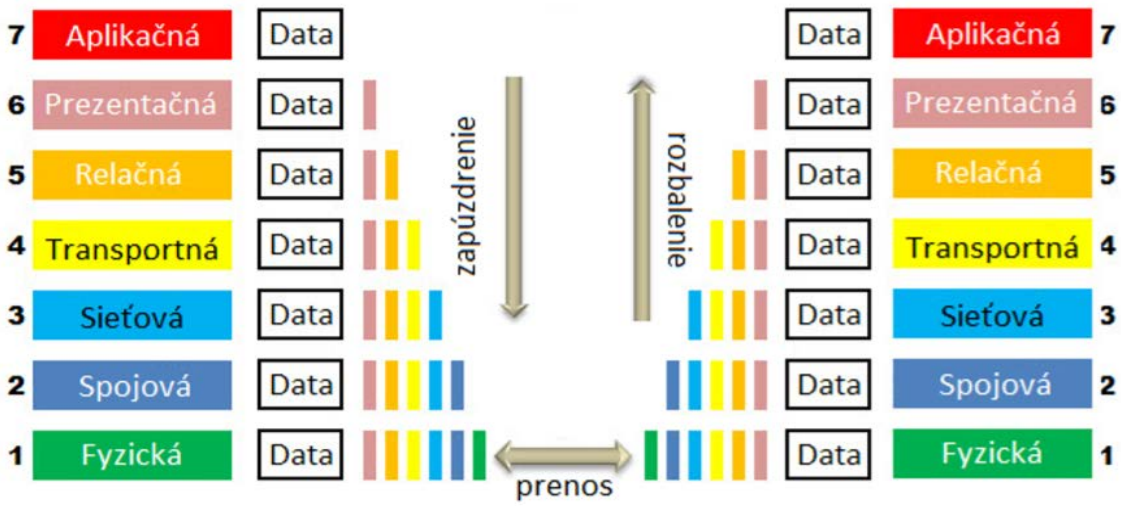


2.1 Vrstvové modely a protokoly

Pri odosielaní dáta prechádzajú cez jednotlivé vrstvy vrstvového modelu, kde sa pripojuje hlavička príslušnej vrstvy. Proces sa nazýva zapuzdrovanie - enkapsutácia (Encapsulation).

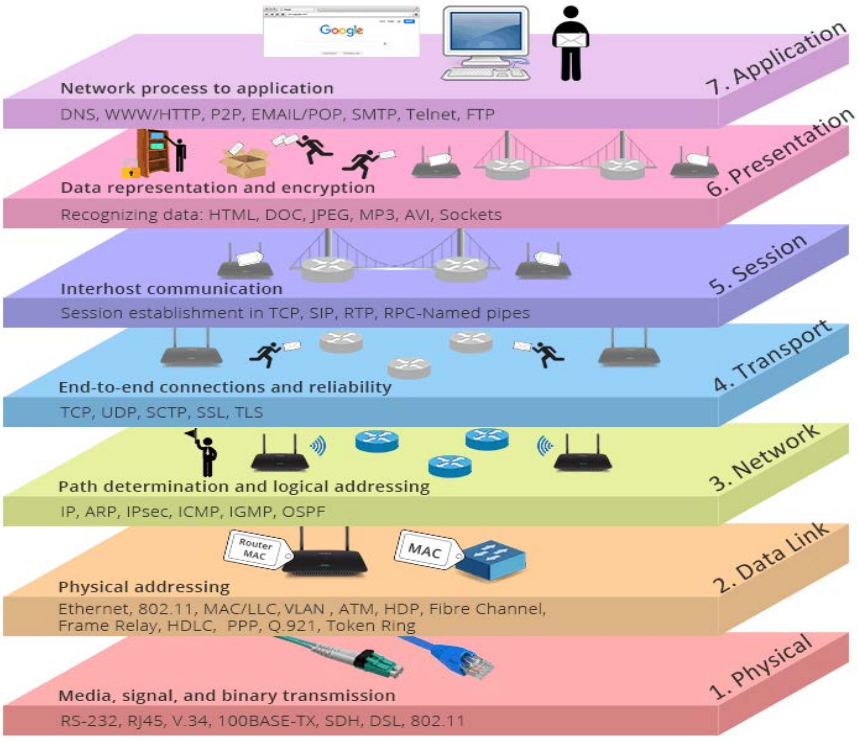
Pri prijímaní dát sa vykonáva odpuzdrovanie – dekapsulácia (Decapsulation). Je opačný proces, kde sa pri prechode vrstvového modelu odpájajú hlavičky príslušnej vrstvy.





Najznámejšie bežné sieťové protokoly

- HTTP - Hypertext Transfer Protocol
- HTTPS - Hypertext Transfer Protocol Secure
- DNS - Domain Name System Protokola
- DHCP - Dynamic Host Configuration Protocol
- FTP - File Transfer Protocol
- TFTP - Trivial File Transfer Protocol
- SMB - Server Message Block
- IMAP - Internet Message Access Protocol



2.2 IoT protokoly

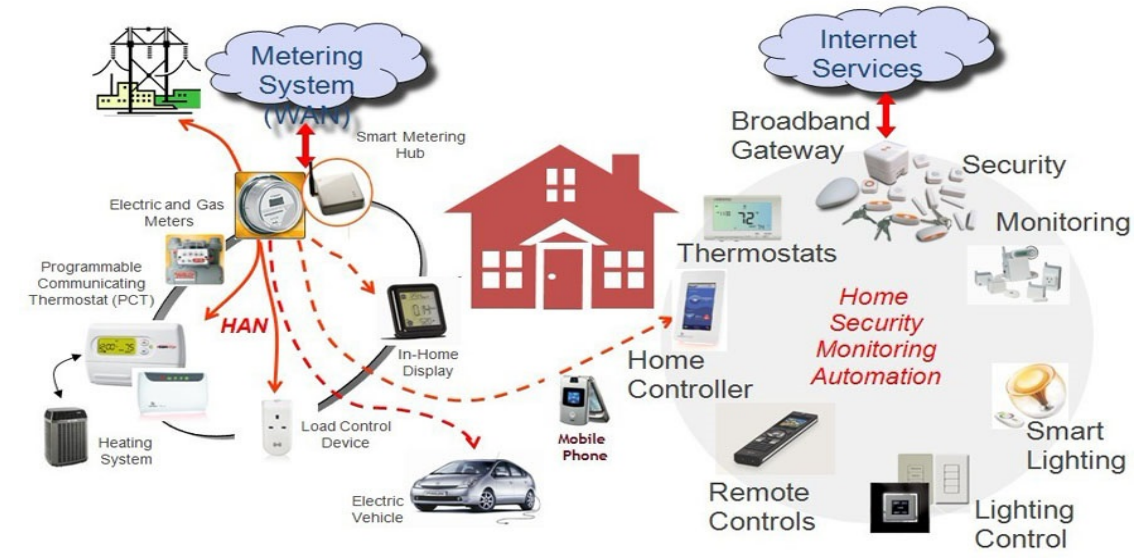
V súčasnosti existuje niekoľko protokolov vhodných pre využitie v prostredí IoT. Fungujú na rôznych vrstvách sieťového modelu OSI:

Vrstva (Layer)	Protokoly (Protocols)
Relačná vrstva (Session)	MQTT, SMQTT, AMQP, CoAP, XMPP, DDS
Sieťová vrstva (Network)	RPL, CORPL, CARP, 6LoWPAN, 6TiSCH, 6Lo, IPv6 over G.9959, IPv6 over Bluetooth Low Energy
Linková vrstva (Data link)	IEEE 802.15.4e, IEEE 802.11 ah, WirelessHART, Z-Wave, Bluetooth Low Energy, Zigbee Smart Energy, DASH7, HomePlug, G.9959, LTE-A, LoRaWAN, Weightless, DECT/ULE

Linková vrstva

ZigBee

Využíva sa na bezdrôtovú dátovú komunikáciu s nízkymi rýchlosťami prenosu dát medzi rôznymi elektronickými zariadeniami v krátkej vzdialenosti s nízkou spotrebou energie. Maximálna prenosová rýchlosť je 250 kb/s. Táto technológia je vhodná pre aplikáciu, pre ktorú je dátový prenos malý a je potrebných veľa zariadení. S pomocou ZigBee protokolu je možné vytvoriť sieť snímačov a pokúsiť sa aplikovať na rôzne monitorovacie a riadiace aplikácie, ako napríklad ovládanie klimatizácie, riadenie osvetlenia, riadenie fyzickej distribúcie, riadenie domu.



Bluetooth LE

Bluetooth LE je charakteristickou črtou špecifikácie Bluetooth 4.0. Je navrhnutý pre aplikácie s extrémne nízkym výkonom, ale zachováva podobnosť s klasickým rozhraním Bluetooth. Vo fyzickej vrstve používa technológia Bluetooth LE aj naďalej adaptačné spektrum šírenia frekvencií (FHSS). Počet kanálov sa zmenšil zo 79 (v klasickom Bluetooth) na 40. Základná rýchlosť Bluetooth LE je 1 Mb / s. Bluetooth LE má z pohľadu pokrytia dosah až niekoľko desiatok metrov.

Near Field Communication (NFC)

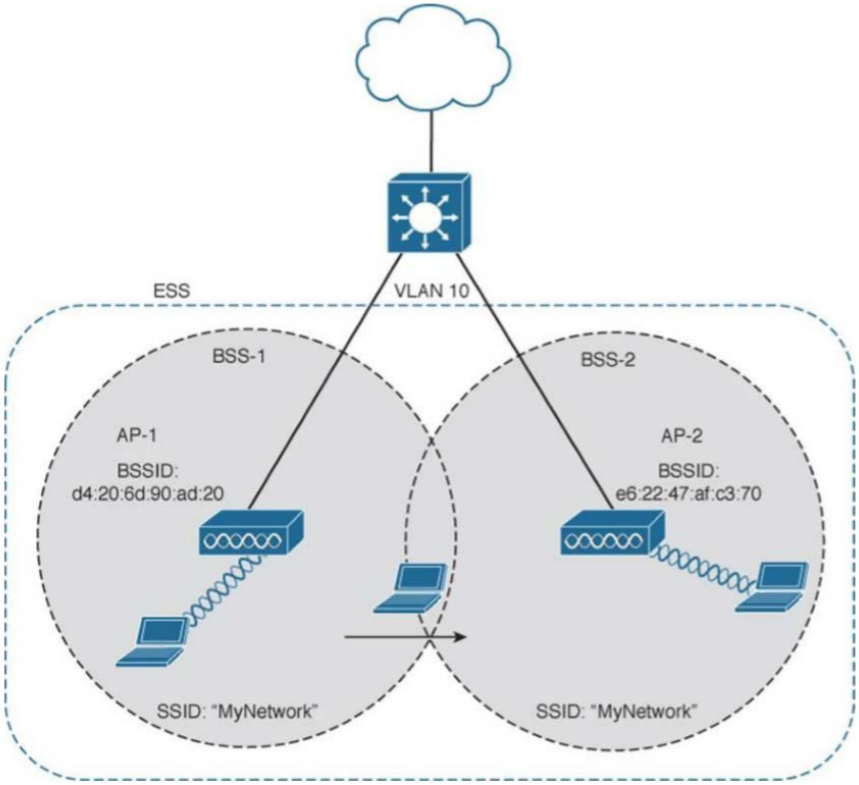
Technológia NFC v súčasnosti umožňuje ľuďom integrovať svoje vernostné karty, kreditné karty do svojich mobilných telefónov. Okrem integrácie týchto kariet do mobilných zariadení prináša technológia NFC aj inovatívne príležitosti pre mobilné komunikácie. Umožňuje dvom používateľom ľahko komunikovať a vymieňať si dáta jednoducho dotykom dvoch mobilných telefónov. Technológia NFC navyše poskytuje schopnosti čítačky NFC pre mobilné telefóny; takže je možné čítať značky RFID (Radio Frequency Identification).

WiFi

Nora IEEE 802.11 rozširuje sieťové štandardy rady 802 na bezdrôtové médium tak, že špecifikuje komunikáciu bezdrôtovej lokálnej siete (WLAN) v pásmach ISM. Prvá verzia bola zverejnená v roku 1997. V nastaveniach infraštruktúry sa bezdrôtové stanice (STA - Station) pripájajú alebo spájajú s prístupovým bodom (AP – Access point). Toto zoskupenie zariadení (STA + AP) sa nazýva základná služba (BSS), kde sa každý STA môže pripojiť k externej sieti (Internetu) prostredníctvom pridruženého AP.

BSS používa identifikačné číslo služby (SSID) na identifikáciu, pričom bežní ľudia poznajú toto SSID skôr pod názvom siete. Viaceré AP je možné pripojiť cez káblový distribučný systém (DS), kde sú rôzne BSS označované ako rozšírená služba (ESS - Extended Service Set) . V scenári, v ktorom BSS používajú rôzne SSID, môže STA zmeniť priradenie do siete s označením SSID, ale musí zmeniť svoju asociáciu s iným AP, čo spôsobí dočasnú stratu spojenia.

Základný identifikátor siete (BSSID) je fyzická adresa (MAC) AP, čo umožňuje STA identifikovať jedinečný BSS AP v ESS.



Sieťová vrstva

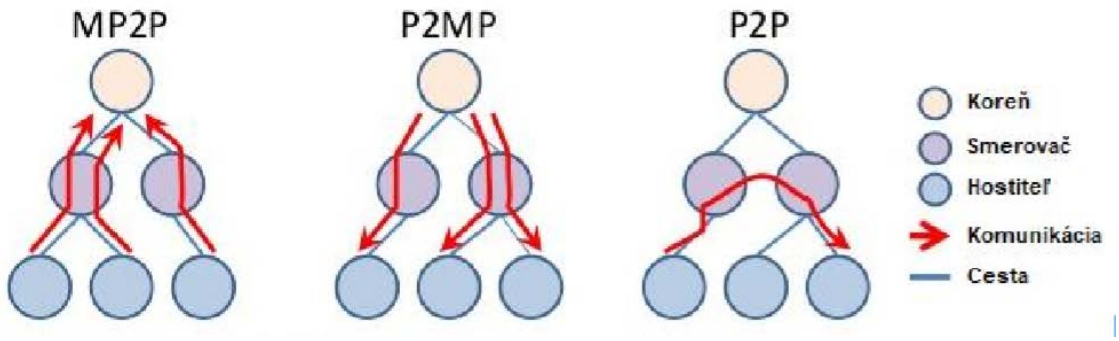
Pre komunikáciu v IoT, boli vyvinuté nové protokoly pre smerovanie komunikácie v sieťach. Dôvodom vytvorenia nových komunikačných protokolov, bola potreba naplniť špecifické potreby, ktoré vznikajú pri komunikácii medzi IoT zariadeniami.

RPL

Protokol chce byť odpoveďou na problémy zariadení napájaných z batérie, ktoré sú pripojené do siete so slabým alebo vypadávajúcim signálom, čo sa prejavuje vysokou mierou straty dát.

RPL môže zahŕňať rôzne druhy dopravných a signalizačných informácií, ktoré sa vymieňajú medzi uzlami, typ informácií závisí od požiadaviek kladených na dátové toky.

- Protokol podporuje prepojenia:
- MP2P (Multipoint to point),
- P2MP (Point to MultiPoint),
- P2P (Point-to-Point).



CROPL

Rozšírením protokolu RPL je kognitívna RPL, označovaný aj ako CROPL. Tento protokol má dve nové modifikácie.

Využíva oportunistické presmerovanie paketov pre výber z pomedzi viacerých ciest k cieľu.

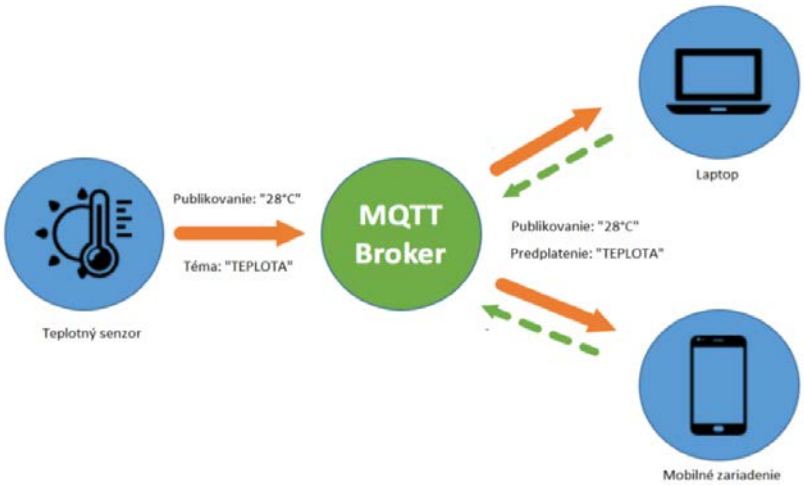
Koordinuje medzi uzlami, aby si vybral najlepší ďalší skok.

Na základe aktualizovaných informácií každý uzol dynamicky aktualizuje svoje susedské priority a vytvára vlastné rozhodovanie o ceste k cieľu.

Relačná vrstva

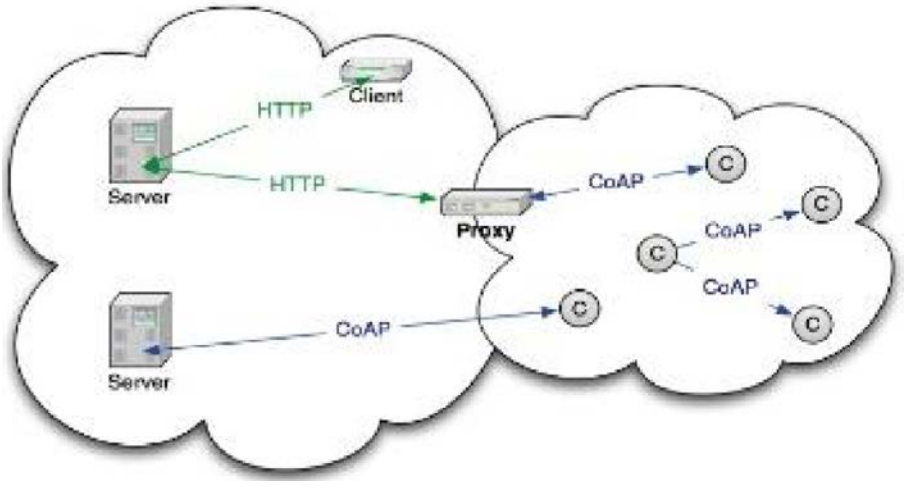
MQTT

Message Queue Telemetry Transport: je komunikačný protokol navrhnutý pre zabezpečovanie asynchrónnej komunikácie medzi aplikáciami a middleware systémom, ktorý beží na vzdialenom serveri. Architektúra sa skladá z vydavateľa (publisher), odoberateľa (subscriber), makléra (broker). Pre zvýšenie bezpečnosti bol vyvinutý protokol SMQTT, čo je Secure MQTT.



CoAP

Constrained Application Protocol: je protokol určený pre zariadenia s obmedzeným prostriedkami. Ide o akúsi odľahčenú verziu protokolu HTTP. CoAP je založený na protokole HTTP a modeli REST (kde sú cieľové informácie načítané zo servera s pomocou URI /URL identifikátorov).



2.3 IoT bezdrôtové komunikačné technológie

V závislosti od aplikácie budú faktory ako dosah, dátové požiadavky, požiadavky na bezpečnosť a napájanie a výdrž batérie diktovať výber jednej alebo nejakej formy kombinácie technológií.

Bluetooth

- Dôležitá komunikačná technológia krátkeho dosahu
- Očakáva sa, že to bude kľúčové najmä pre nositeľné produkty, opäť sa pripojí k internetu vecí, aj keď v mnohých prípadoch pravdepodobne cez smartfón.
- Nový Bluetooth Low-Energy (BLE) – alebo Bluetooth Smart, ako je teraz označovaný – je významným protokolom pre aplikácie internetu vecí.
- Dôležité je, že hoci ponúka podobný dosah ako Bluetooth, bol navrhnutý tak, aby ponúkal výrazne nižšiu spotrebu energie.
- Vzhľadom na svoju rozsiahlu integráciu do smartfónov a mnohých ďalších mobilných zariadení má určite veľkú výhodu v kontexte osobnejších zariadení oproti mnohým konkurenčným technológiám.
- Podľa Bluetooth SIG sa očakáva, že viac ako 90 percent smartfónov s podporou Bluetooth, vrátane modelov so systémom iOS, Android a Windows, bude do roku 2021 „Smart Ready“.

- Štandard: Špecifikácia jadra Bluetooth 4.2Frekvencia: 2,4 GHz (ISM)Dosah: 50-150 m (Smart/BLE)Dátové rýchlosti: 1 Mbps (Smart/BLE)

ZigBee

- Použitie možno tradične viac v priemyselnom prostredí.
- Založené na protokole IEEE802.15.4, čo je priemyselná štandardná bezdrôtová sieťová technológia fungujúca na frekvencii 2,4 GHz zameraná na aplikácie, ktoré si vyžadujú relatívne zriedkavé výmeny údajov pri nízkych rýchlostiach údajov v obmedzenom priestore a v dosahu 100 m, napríklad v domácnosti alebo budove.
- Štandard: ZigBee 3.0 založený na IEEE802.15.4Frekvencia: 2,4 GHz
- Dosah: 10-100mDátové rýchlosti: 250 kbps

Z-Wave

- Nízkoenergetická RF komunikačná technológia, ktorá je primárne navrhnutá pre domácu automatizáciu pre produkty, ako sú ovládače lámp a senzory a mnohé iné.
- Štandard: Z-Wave Alliance ZAD12837 / ITU-T G.9959Frekvencia: 900 MHz (ISM)Dosah: 30mDátové rýchlosti: 9,6/40/100 kbit/s

6LoWPAN

- Kľúčovou technológiou založenou na IP (Internet Protocol) je 6LowPAN (IPv6 Low-power wireless Personal Area Network). Namiesto technológie aplikačných protokolov internetu vecí, ako je Bluetooth alebo ZigBee, je 6LowPAN sieťový protokol, ktorý definuje mechanizmy zapuzdrenia a kompresie hlavičiek.
- Štandard: RFC6282Frekvencia: (prispôsobená a používaná v rôznych ďalších sieťových médiách vrátane Bluetooth Smart (2,4 GHz) alebo ZigBee alebo nízkoenergetického RF (menej ako 1 GHz)
- Rozsah: „neurčité“
- Dátové sadzby: „neurčité“

Thread

- Úplne nový sieťový protokol IPv6 na báze IP zameraný na prostredie domácej automatizáciez aplikačného hľadiska je primárne navrhnutý ako doplnok k WiFi, pretože uznáva, že aj keď je WiFi dobré pre mnohé spotrebiteľské zariadenia, má obmedzenia pre použitie v nastavení domácej automatizácie..

- Štandard: vlákno, založené na IEEE802.15.4 a 6LowPANFrekvencia: 2,4 GHz (ISM)Rozsah: N/ADátové sadzby: N/A

WiFi

- Štandard: Na základe 802.11n (dnes najbežnejšie používanie v domácnostiach)Frekvencie: pásma 2,4 GHz a 5 GHzDosah: približne 50 mDátové rýchlosti: maximálne 600 Mb/s, ale bežnejšie je 150 – 200 Mb/s, v závislosti od použitej kanálovej frekvencie a počtu antén (najnovší štandard 802.11-ac by mal ponúkať 500 Mb/s až 1 Gb/s)

Cellular

- Štandard: Na základe 802.11n (dnes najbežnejšie používanie v domácnostiach)Každá aplikácia IoT, ktorá vyžaduje prevádzku na väčšie vzdialenosti, môže využívať možnosti mobilnej komunikácie GSM/3G/4G. Aj keď je mobilný telefón schopný odosielať veľké množstvo údajov, najmä pre 4G, náklady a spotreba energie budú pre mnohé aplikácie príliš vysoké, ale môžu byť ideálne pre projekty s nízkou šírkou pásma založené na senzoroch, ktoré budú odosielať veľmi nízke množstvo dát cez internet.

- Štandard: GSM/GPRS/EDGE (2G), UMTS/HSPA (3G), LTE (4G)Frekvencie: 900/1800/1900/2100MHzDosah: max. 35 km pre GSM; 200 km max pre HSPADátové rýchlosti (typické sťahovanie): 35 – 170 kb/s (GPRS), 120 – 384 kb/s (EDGE), 384 kb/s – 2 Mb/s (UMTS), 600 kb/s – 10 Mb/s (HSPA), 3 – 10 Mb/s (LTE)

NFC

- Štandard: GSM/GPRS/EDGE (2G), UMTS/HSPA (3G), LTE (4G)Frekvencie: 900/1800/1900/2100MHzDosah: max. 35 km pre GSM; 200 km max pre HSPANFC (Near Field Communication) je technológia, ktorá umožňuje jednoduché a bezpečné obojsmerné interakcie medzi elektronickými zariadeniami a je obzvlášť vhodná pre smartfóny, ktorá umožňuje spotrebiteľom vykonávať bezkontaktné platobné transakcie, pristupovať k digitálnemu obsahu a pripájať elektronické zariadenia.

- Norma: ISO/IEC 18000-3Frekvencia: 13,56 MHz (ISM)Rozsah: 10 cmDátové rýchlosti: 100–420 kbps

Sigfox

- Alternatívnou širokopásmovou technológiou je Sigfox, ktorá z hľadiska dosahu patrí medzi Wi-Fi a mobilné siete. Na prenos dát vo veľmi úzkom spektre do a z pripojených objektov využíva ISM pásma, ktoré je možné voľne využívať bez nutnosti získavania licencií.

- Myšlienka Sigfox je taká, že pre mnohé aplikácie M2M, ktoré bežia na malú batériu a vyžadujú len nízku úroveň prenosu dát, je dosah WiFi príliš krátky, zatiaľ čo mobilné pripojenie je príliš drahé a tiež spotrebúva príliš veľa energie.Sigfox používa technológiu nazývanú Ultra Narrow Band (UNB) a je navrhnutá len tak, aby zvládala nízke rýchlosti prenosu dát 10 až 1 000 bitov za sekundu. Spotrebuje iba 50 mikrowattov v porovnaní s 5 000 mikrowattmi pre mobilnú komunikáciu, alebo dokáže poskytnúť typickú pohotovostnú dobu 20 rokov s 2,5 Ah batériou, zatiaľ čo pre mobilnú komunikáciu je to len 0,2 roka.

- Štandard: SigfoxFrekvencia: 900 MHzDosah: 30-50 km (vidiecke prostredie), 3-10 km (mestské prostredie)Dátové rýchlosti: 10-1000 bps

Neul

- Koncepcia podobná Sigfoxu a prevádzka v pásme pod 1 GHz, Neul využíva veľmi malé časti spektra TV White Space na poskytovanie vysokej

škálovateľnosti, vysokého pokrytia, nízkej spotreby a lacných bezdrôtových sietí..

- Dátové rýchlosti môžu byť čokoľvek od niekoľkých bitov za sekundu až po 100 kbps cez to isté prepojenie; a zariadenia dokážu spotrebovať už od 20 do 30 mA z 2xAA batérií, čo znamená 10 až 15 rokov v teréne.

- Štandard: NeulFrekvencia: 900 MHz (ISM), 458 MHz (UK), 470-790 MHz (White Space)Dojazd: 10 kmDátové rýchlosti: Niekoľko bps až 100 kbps

LoRaWAN

- Opäť, podobne ako v niektorých ohľadoch Sigfox a Neul, sa LoRaWAN zameriava na aplikácie rozľahlej siete (WAN) a je navrhnutý tak, aby poskytoval nízkoenergetické siete WAN s funkciami špecificky potrebnými na podporu lacnej mobilnej bezpečnej obojsmernej komunikácie v IoT, M2M a inteligentné mesto a priemyselné aplikácie.

- Optimalizované pre nízku spotrebu energie a podporujúce veľké siete s miliónmi a miliónmi zariadení, prenosové rýchlosti sa pohybujú od 0,3 kbps do 50 kbps.

- Štandard: LoRaWANFrekvencia: RôzneDosah: 2-5 km (mestské prostredie), 15 km (prímestské prostredie)Dátové rýchlosti: 0,3-50 kbps.

2.4 Komunikačný subsystém

Existuje mnoho možností (protokolov) ako komunikovať s externým úložiskom dát:

MQTT protokol (Message Queuing Telemetry Transport)

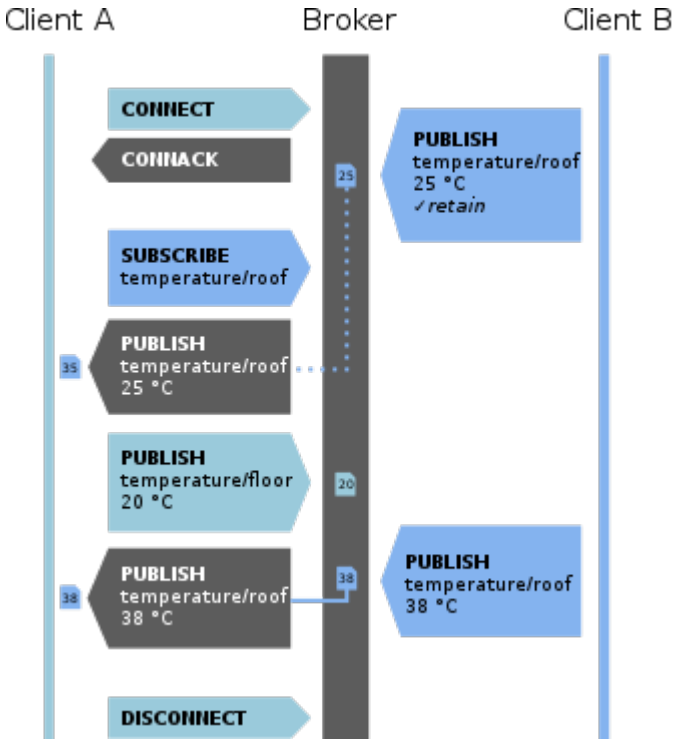
- Definovaný v ISO štandarde ISO/IEC PRF 20922

- Aplikačný protokol (TCP/IP), ktorý využíva metódy PUBLISH a SUBSCRIBE

- Veľmi jednoduchý protokol s minimálnymi HW nárokmi (prenosová rýchlosť (veľkosť záhlavia 2B), výkon procesora a veľkosť pamäte).

- Zabezpečuje komunikáciu klienta so vzdialeným serverom (broker)

- Klient môže byť odosielateľom informácie (publisher), ale aj prijímateľom informácie (subscriber).



HTTP, resp. HTTPS protokol

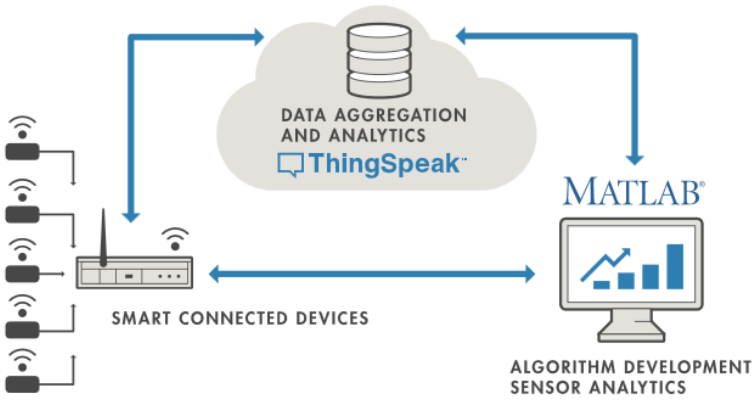
- Využitie štandardného protokolu pre potreby IoT
- Využívané najmä pri komunikácii s cloudovými službami (ThingSpeak, FireBase)
- Zložitejší protokol s vyššími nárokmi na pamäť, procesor a rýchlosť komunikačnej linky

Iné protokoly:

MQTT-SN (MQTT For Sensor Networks), AMQP (Advanced Message Queuing Protocol), DDS (Data-Distribution Service for Real-Time Systems), IBM MessageSight, STOMP - The Simple Text Oriented Messaging Protocol, XMPP (Extensible Messaging and Presence Protocol), LLAP (lightweight local automation protocol), ...

2.5 ThingSpeak

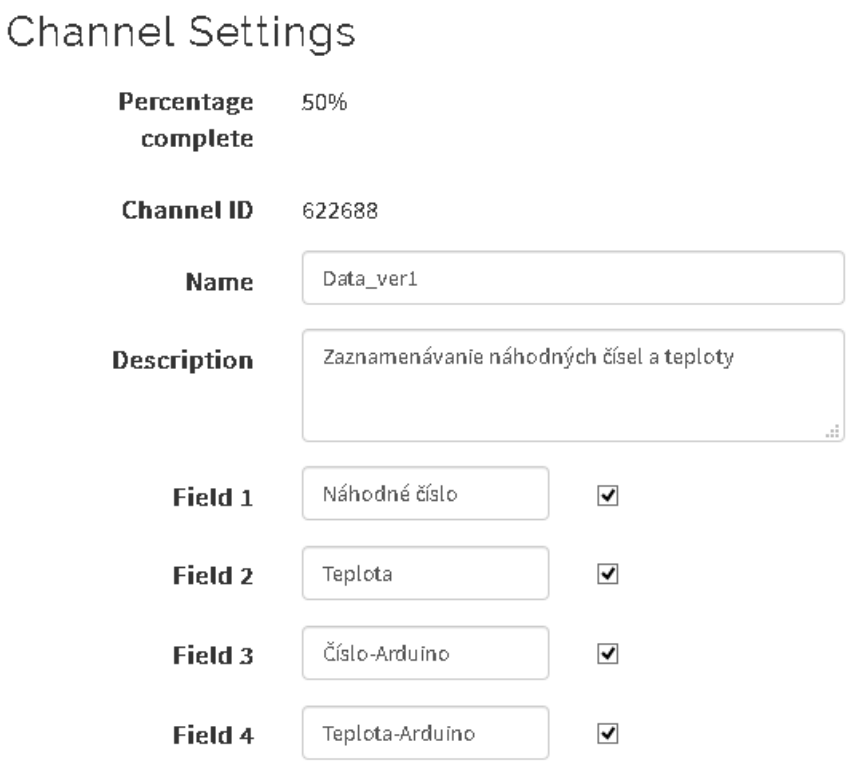
ThingSpeak je cloudová služba umožňujúca zber dát a ich následné prezentovanie vo forme grafov. Pre neplatičov (vo verzii zadarmo) ponúka obmedzenia typu počet kanálov (skupiny dát) a pod. čo však pre demonštráčné a výučbové účely bohato stačí. Táto platforma ponúka API, vďaka ktorému je ju možné implementovať do rôznych platforiem ako Raspberry Pi, Arduino, ESP 32 atď.



Postup práce s ThinkSpeak

- Registrácia na portáli www.thingspeak.com
- Prihlásenie sa po registrácii na portál
- Vytvorenie vlastného kanála
- Získanie informácií o kanáli (channel)

Channel ID: Slúži pre identifikáciu kanála



Write API Key: Tajný kľúč používaný na zápis údajov. V prípade jeho prezradenia možno vygenerovať nový kľúč.

Read API Key: Tajný kľúč používaný na čítanie údajov. V prípade jeho prezradenia možno vygenerovať nový kľúč.

Príklad ukladania dát z Raspberry Pi do sieťového úložiska ThingSpeak

Zdrojový kód v Pythone ukladajúci náhodné číslo do úložiska

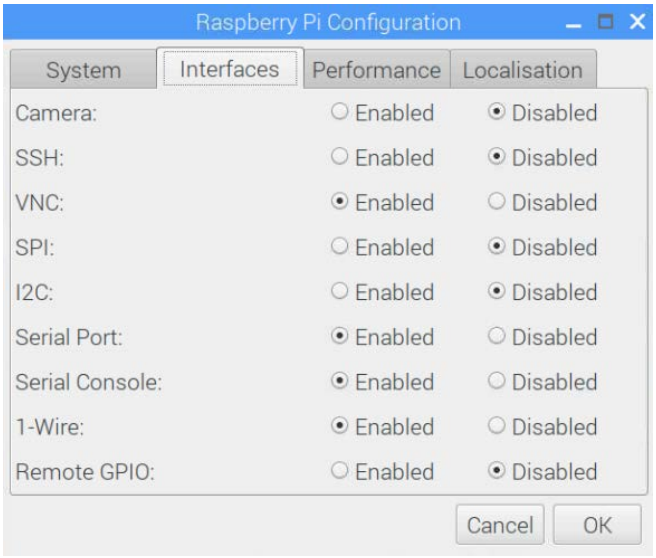
```
import sys # prilinkovanie kniznic / funkcii

import urllib.request

from time import sleep

import random

mojeAPI = '8LWWBU4TTPECU67S' # definovanie kluca pre zapis
        # na serveri thingspeak.com
```

```
# def. URL servera
webURL = 'https://api.thingspeak.com/update?api_key=%s' % mojeAPI
while True: # nekonecny cyklus
    hodnota = random.gauss(5,10) # generovanie nahodnej hodnoty
    # odoslanie dat na server
    spojenie = urllib.request.urlopen(webURL + '&field1=%s' % hodnota)
    spojenie.close() # ukoncenie spojenia
    sleep(15)  # Minimalna doba pre upgrade
```

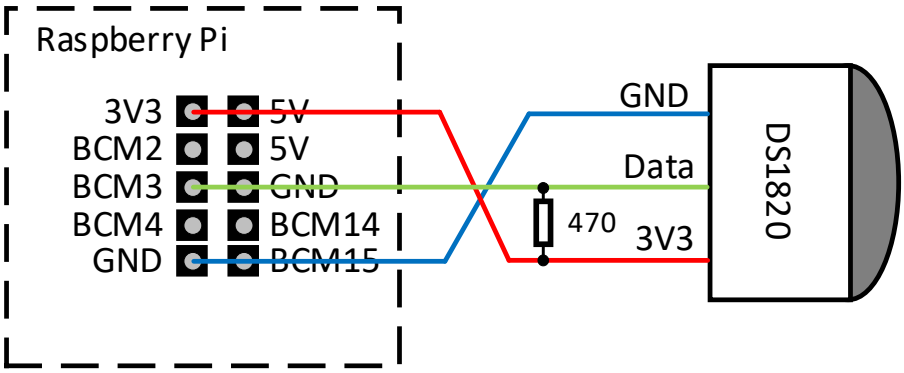
Povolenie 1-wire rozhrania Menu -> Preferences -> Raspberry Pi Configuration

Reštart systému

sudo reboot



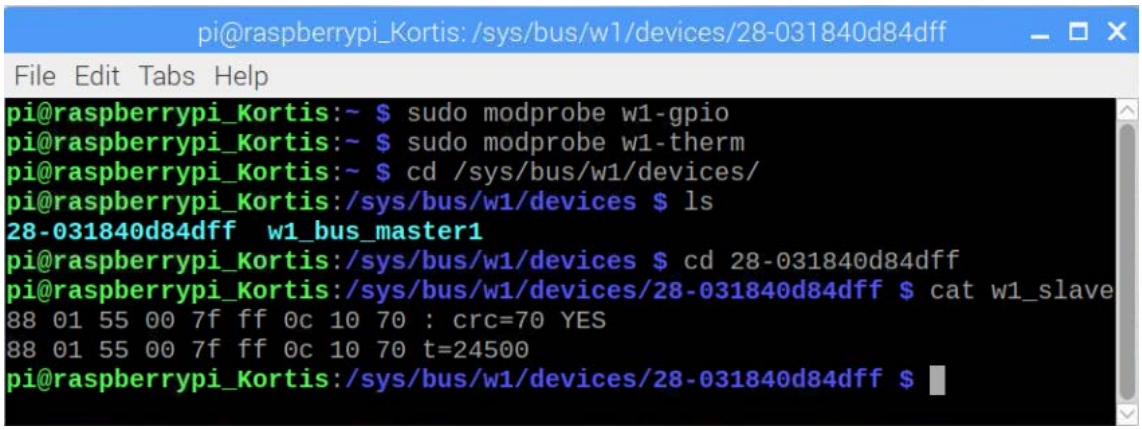
Zapojenie teplotného senzora DS1820



Po pripojení modulov do jadra systému:

sudo modprobe w1-gpio sudo modprobe w1-therm

Možno v adresári **/sys/bus/w1/devices** nájsť podadresár **28-XXXX**, kde XXXX je adresa senzora. V podadresári sa nachádza súbor **w1-slave**, ktorý obsahuje aj teplotu zo senzora.



Zdrojový kód v Pythone

import sys # prilinkovanie kniznic / funkcii

import urllib.request

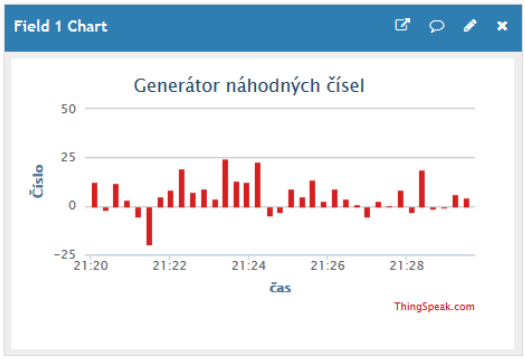
from time import sleep

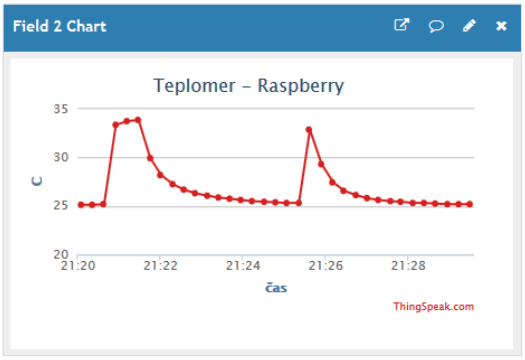
import random

```
        import RPi.GPIO as GPIO

        import os

        import glob

        mojeAPI = '8LWWBU4TTPECU67S' # definovanie kluca pre zapis
# na serveri thingspeak.com

        # def. URL servera

        webURL = 'https://api.thingspeak.com/update?api_key=%s' % mojeAPI

        os.system('modprobe w1-gpio')

        os.system('modprobe w1-therm')

        base_dir = '/sys/bus/w1/devices/'

        device_folder = glob.glob(base_dir + '28*')[0]

        device_file = device_folder + '/w1_slave'


        def read_temp_raw():

            f = open(device_file, 'r')

            lines = f.readlines()

            f.close()

            return lines


        def read_temp():

            lines = read_temp_raw()

            while lines[0].strip()[-3:] != 'YES':
```

```
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        return temp_c
while True: # nekonecny cyklus
    hodnota = random.gauss(5,10) # generovanie nahodnej hodnoty s
                        # normalnym rozdelenim
teplota = read_temp()
    # odoslanie dat na server
    spojenie = urllib.request.urlopen(webURL + '&field1=%s' % hodnota +
        '&field2=%s' % teplota)

    #kontrolny vypis na obrazovku
    print (spojenie.read()) # vypis odpovede zo servera na obrazovku

    spojenie.close() # ukoncenie spojenia
    sleep(15)
```

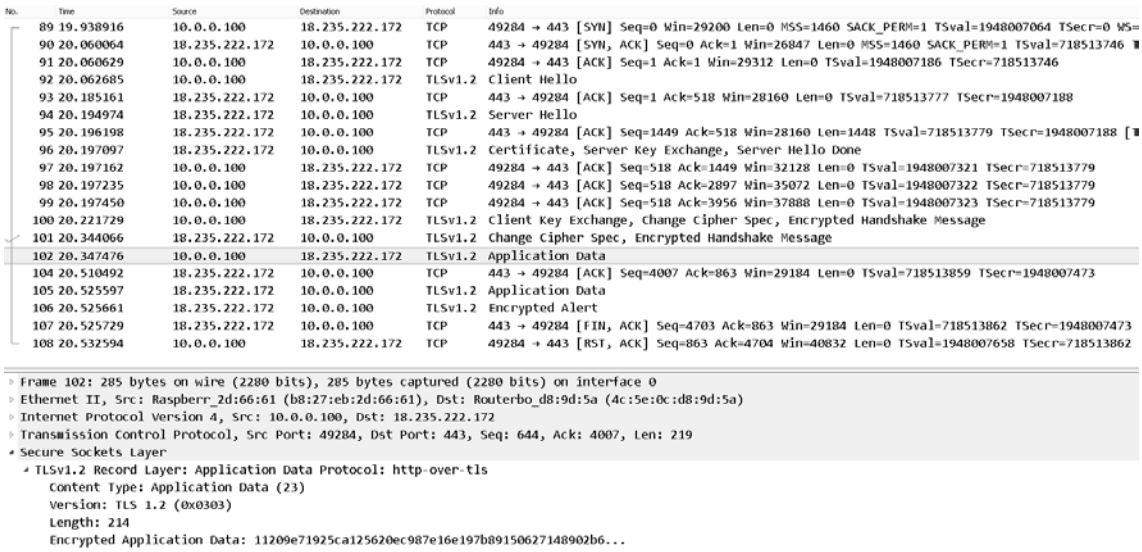
Grafy so zaznamenanými údajmi



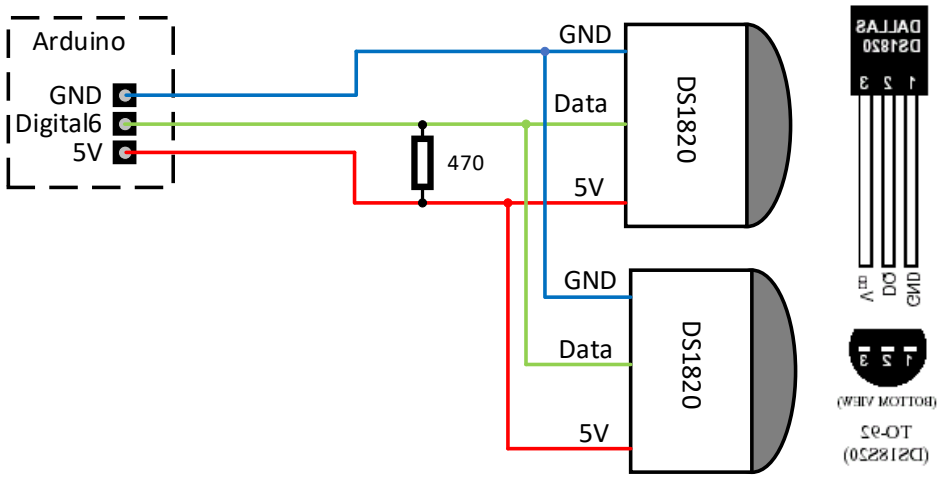


Paketová komunikácia zodpovedajúca jednej transakcii s úložiskom ThingSpeak

(HTTP komunikácia je šifrovaná - TLS v 1.2 (HTTP over TLS))



Získanie teploty zo senzora DS1820 pomocou Arduina



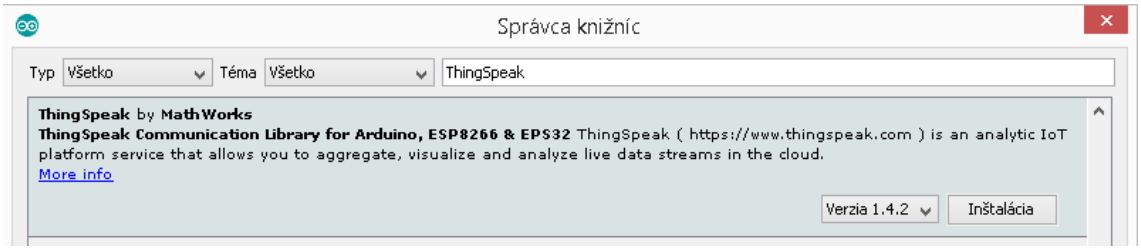
Inštalácia knižnice ThingSpeak do vývojového prostredia:

Projekt –> Zahrnúť knižnice –> Spravovať knižnice

Je vhodné sa inšpirovať hotovými príkladmi

Súbor –> Príklady –> ThingSpeak

ďalej napr. –> ArduinoEthernet –> WriteMultipleFields



Program pre prácu s 1-wire modulom

```
#include <OneWire.h>
#include <DallasTemperature.h>

// PIN pre pripojenie teplotneho senzora 18B20 od fy Dallas
#define ONE_WIRE_BUS 6

// Vytvorenie instancie pre komunikaciu s lubovolnym 1-wire zariadenim
// nielen s teplotnym senzorom 18B20
OneWire oneWire(ONE_WIRE_BUS);

// Vytvorenie novej instancie pre teplotny senzor odkazujucej sa na instanciu oneWire
DallasTemperature sensors(&oneWire);

void setup(void) {
  Serial.begin(9600);
  sensors.begin();
}

void loop(void) {
  Serial.print("Nacitavam teplotu zo senzorov ...   ");
  sensors.requestTemperatures(); // Zaslanie prikazu pre zistenie teploty zo senzorov
  Serial.println("Akcia dokoncena uspesne.");

  // Nacitam teplotu z prveho senzora s indexom 0
  Serial.print("Teplota: ");
  Serial.println(sensors.getTempCByIndex(0));

  delay(1000);
}
```

Program pre prácu s ThingSpeak

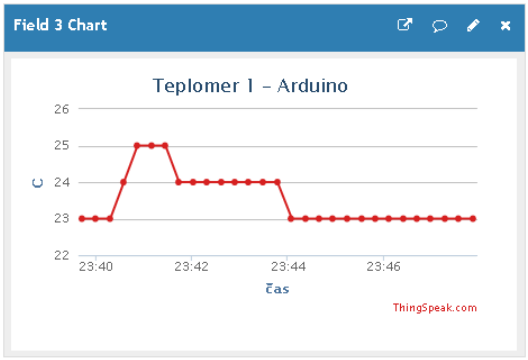
```
#include <SPI.h>
#include <Ethernet.h>
#include "secrets.h"

#include <OneWire.h>
#include <DallasTemperature.h>


byte mac[] = { 0x00, 0x01, 0x02, 0x03, 0x40, 0x4F };

// Set the static IP address to use if the DHCP fails to assign
IPAddress ip(10, 0, 0, 111);
IPAddress myDns(8, 8, 8, 8);

EthernetClient client;

unsigned long myChannelNumber = SECRET_CH_ID;
const char * myWriteAPIKey = SECRET_WRITE_APIKEY;

int number3, number4;

// PIN pre pripojenie teplotneho senzora 18B20 od fy Dallas
#define ONE_WIRE_BUS 6

// Vytvorenie instancie pre komunikaciu s lubovolnym 1-wire zariadenim
// nielen s teplotnym senzorom 18B20
OneWire oneWire(ONE_WIRE_BUS);

// Vytvorenie novej instancie pre teplotny senzor odkazujucej sa na instanciu oneWire
DallasTemperature sensors(&oneWire);


void setup() {
  Serial.begin(9600);  //Initialize serial

  // start the Ethernet connection:
  Serial.println("Initialize Ethernet with DHCP:");
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Failed to configure Ethernet using DHCP");
    Serial.println(Ethernet.localIP());
    // Check for Ethernet hardware present
    if (Ethernet.hardwareStatus() == EthernetNoHardware) {
      Serial.println("Ethernet shield was not found.  Sorry, can't run without hardware. :(");
      while (true) {
        delay(1); // do nothing, no point running without Ethernet hardware
      }
    }
    if (Ethernet.linkStatus() == LinkOFF) {
      Serial.println("Ethernet cable is not connected.");
    }
    // try to congifure using IP address instead of DHCP:
    Ethernet.begin(mac, ip, myDns);
    Serial.println(Ethernet.localIP());
  } else {
    Serial.print("  DHCP assigned IP ");
    Serial.println(Ethernet.localIP());
  }
```

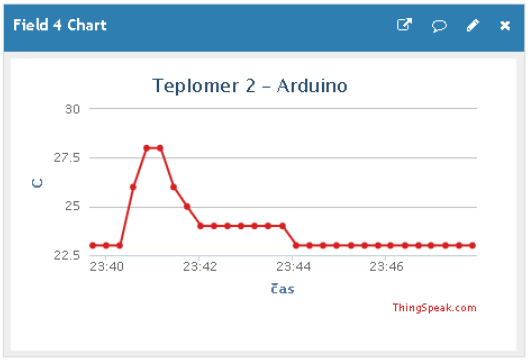
```
  // give the Ethernet shield a second to initialize:
  delay(1000);

  ThingSpeak.begin(client);  // Initialize ThingSpeak

  //sensors.begin();

    uint8_t adresa[8];

  //Serial.begin(9600);
  int PocetSenzorov = 0;
  sensors.begin();
  while (oneWire.search(adresa)) {
    PocetSenzorov++;
  }
  Serial.println("Pocet senzorov: ");
  Serial.print(PocetSenzorov);
  Serial.println("");
}

void loop() {
  sensors.requestTemperatures(); // Zaslanie prikazu pre zistenie teploty zo senzorov
  number3 = sensors.getTempCByIndex(0);
  number4 = sensors.getTempCByIndex(1);

  Serial.println(number3);
  Serial.println(number4);
  // set the fields with the values
  ThingSpeak.setField(3, number3);
  ThingSpeak.setField(4, number4);

  // write to the ThingSpeak channel
  int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
  if(x == 200){
    Serial.println("Channel update successful.");
  }
  else{
    Serial.println("Problem updating channel. HTTP error code " + String(x));
  }

  delay(15000); // Wait 15 seconds to update the channel again
}
```

Súbor secrets.h obsahuje potrebné identifikátory pre prístup k službe ThingSpeak

```
// Use this file to store all of the private credentials
// and connection details

// Enter a MAC address for your controller below.
// Newer Ethernet shields have a MAC address printed on a sticker on the shield
#define SECRET_MAC {0x90, 0xA2, 0xDA, 0x10, 0x40, 0x4F}

//#define SECRET_CH_ID 000000     // replace 0000000 with your channel number
//#define SECRET_WRITE_APIKEY "XYZ"   // replace XYZ with your channel write API Key

#define SECRET_CH_ID 622688
#define SECRET_WRITE_APIKEY "8LWWBU4TTPECU67SK"
```

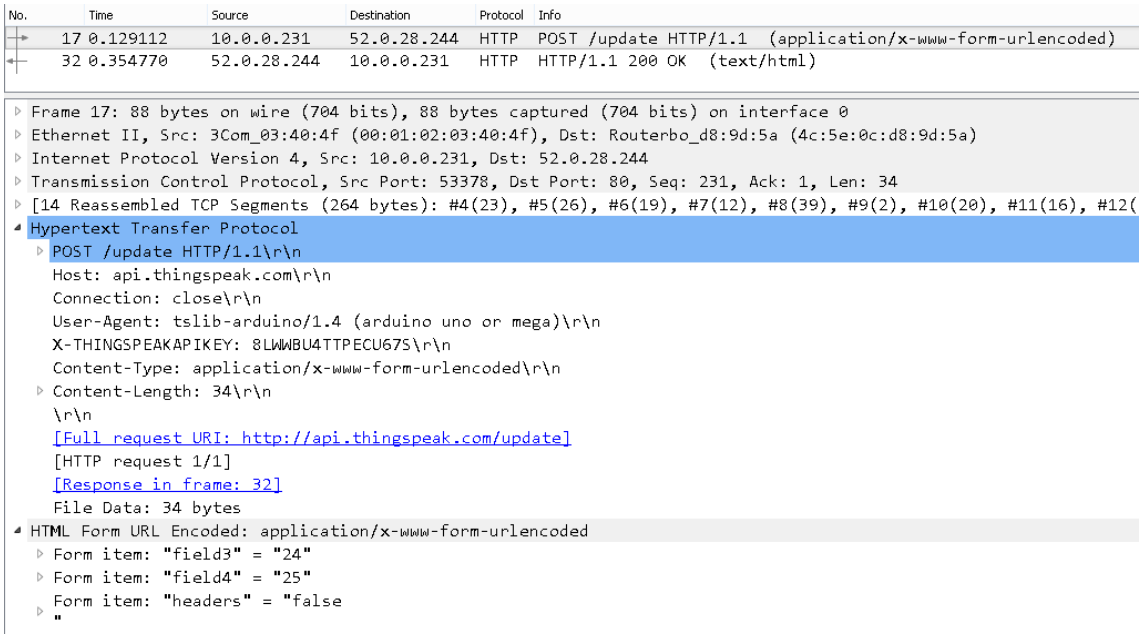
Grafy so zaznamenanými údajmi





Paketová komunikácia zodpovedajúca jednej transakcii s úložiskom ThingSpeak

(HTTP komunikácia nie je šifrovaná)




```
Status: 200 OK
X-Frame-Options: ALLOWALL
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, PUT, OPTIONS, DELETE, PATCH
Access-Control-Allow-Headers: origin, content-type, X-Requested-With
Access-Control-Max-Age: 1800
ETag: "c9f0f895fb98ab9159f51fd0297e236d"
Cache-Control: max-age=0, private, must-revalidate
Set-Cookie: request_method=POST; path=/
X-Request-Id: 3c7547d9-f2dd-4946-8ed7-3ee173d79eeb
X-Runtime: 0.103413
X-Powered-By: Phusion Passenger 4.0.57
Date: Mon, 12 Nov 2018 22:41:45 GMT
Server: nginx/1.9.3 + Phusion Passenger 4.0.57
```

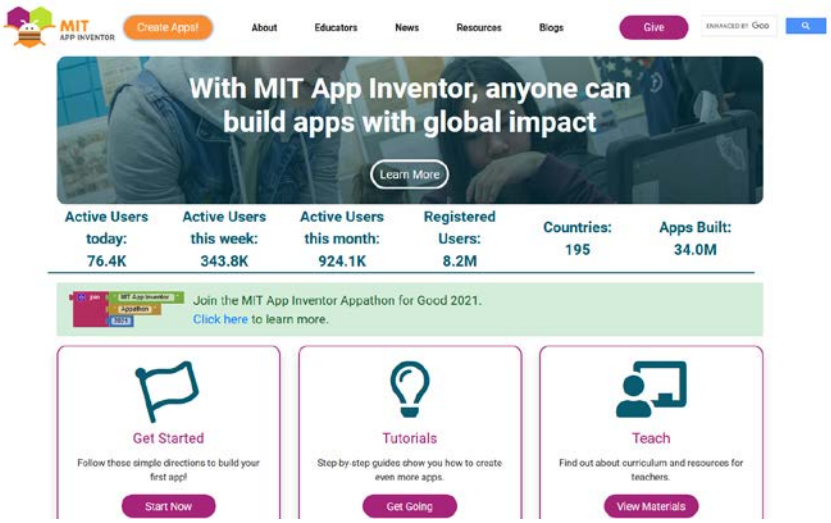
3 MIT APP INVENTOR

3.1 Charakteristika:

- vývojové prostredie a grafický programovací jazyk pre mobilné telefóny
- používa sa pri výuke programovania
- možno v ňom vyvíjať aj plnohodnotné aplikácie, ktoré je možné zverejniť na Google Play
- je podporovaný len operačný systém Android
- prostredie App Invetoru je v internetovom prehliadači
- zdrojový kód sa ukladá v cloude

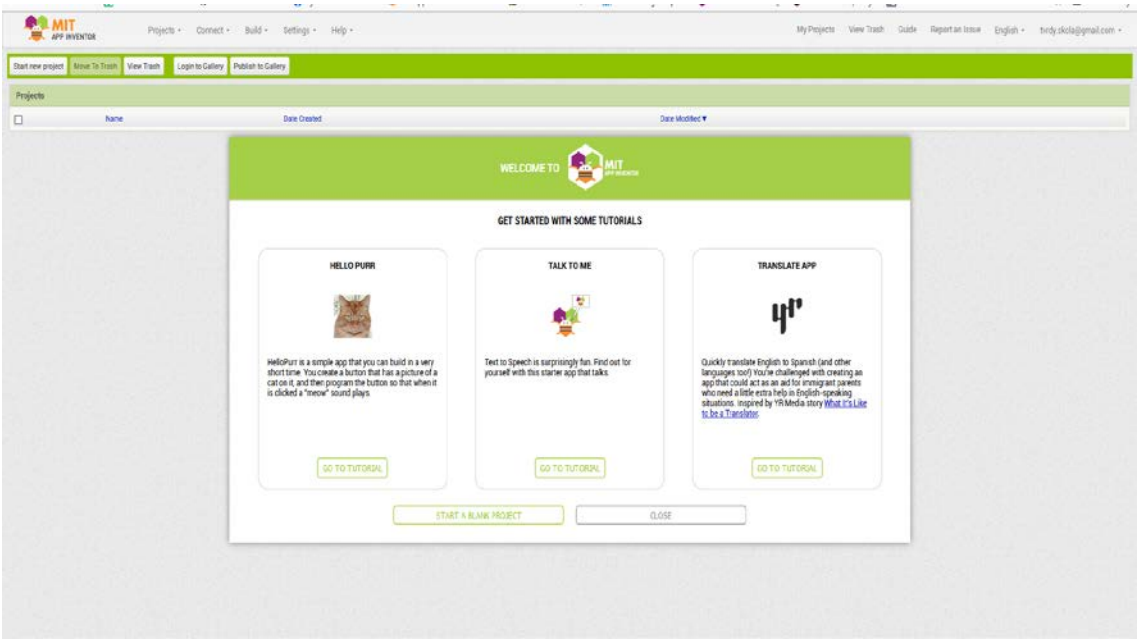
Spustenie vývojového prostredia priamo v internetovom prehliadači na stránke:

https://appinventor.mit.edu/

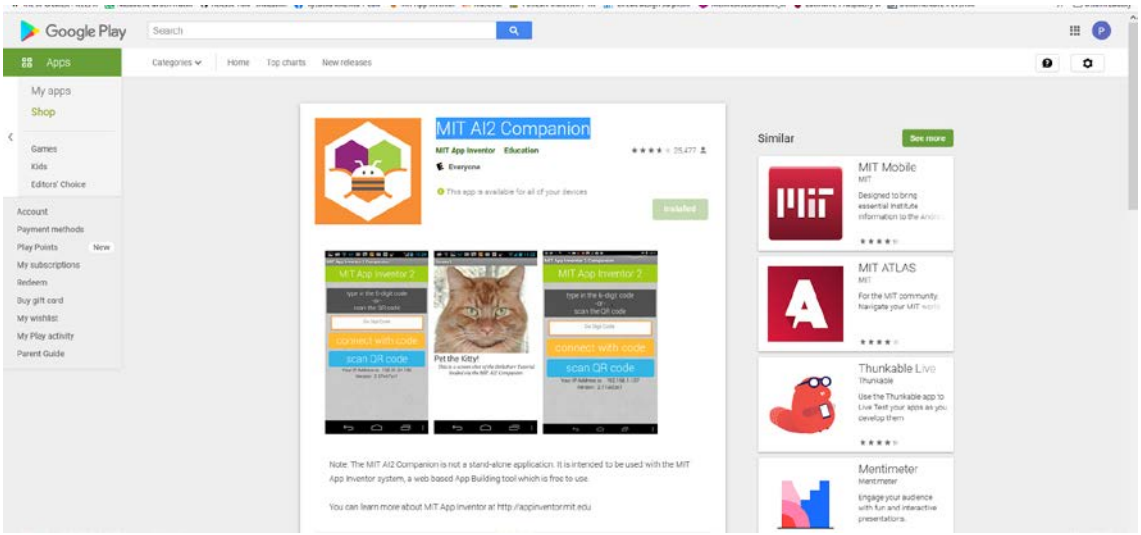


Nasleduje prihlasovanie cez účet google s následným odsúhlasením podmienok používania.

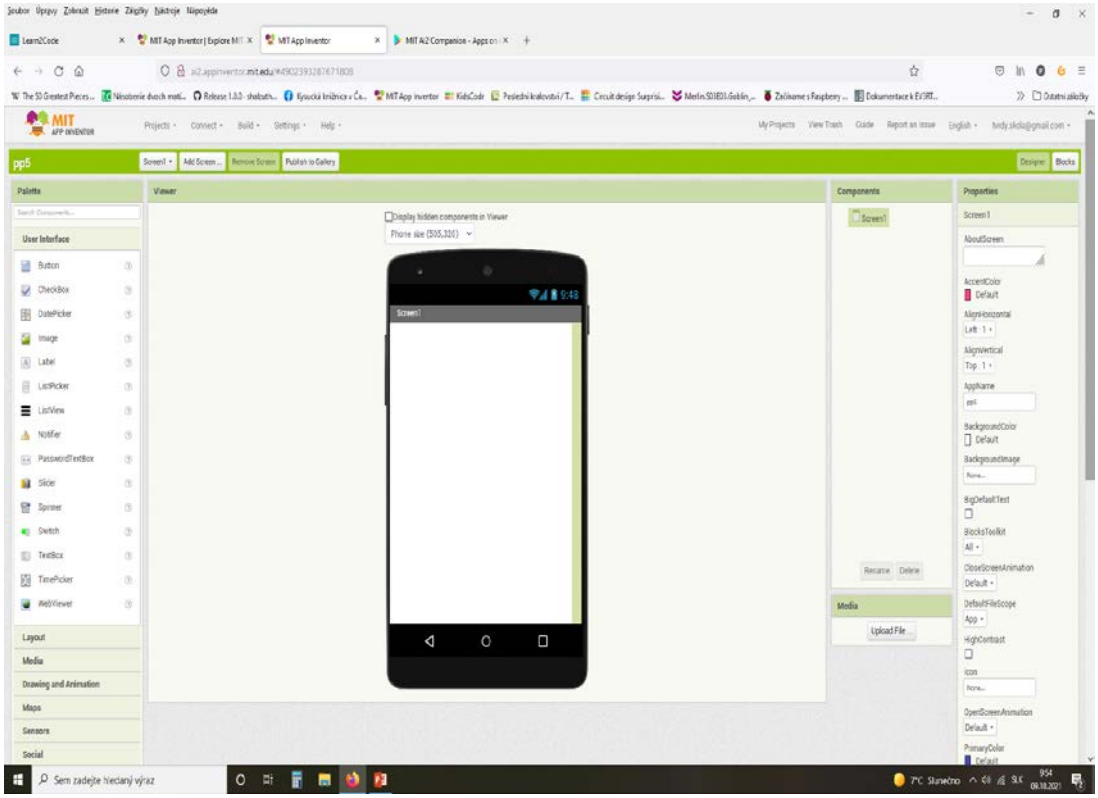
Na nasledujúcom obrázku je zobrazené spustené vývojové prostredie.

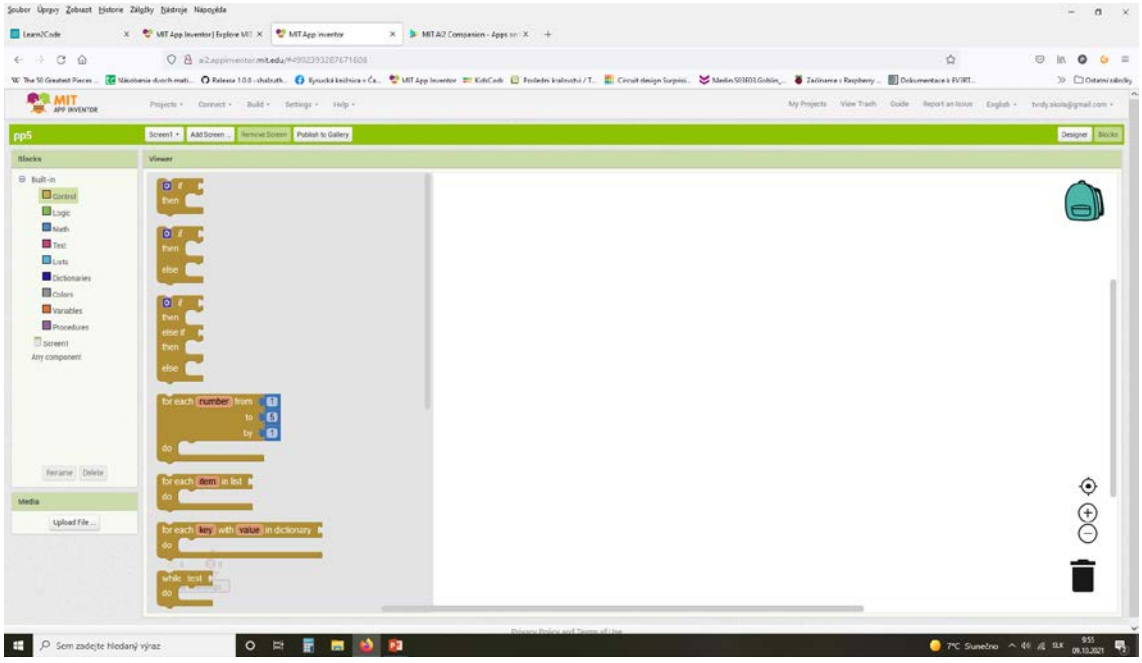


Testovanie aplikácie – priamo v mobilnom telefóne po nainštalovaní aplikácie cez google play - **MIT AI2 Companion** (prípadne simulácia v PC).



- Vývojové prostredie má dve základné obrazovky:
    - Designer - návrh dizajnu aplikácie
    - Blocks - algoritmická časť aplikácie, vkladanie blokov zdrojového kódu





Tvorba aplikácie

1. Projects – Start new project.

2. Zadať názov projektu a potvrdiť.

3. Tvorba dizajnu aplikácie – čo bude na jednotlivých obrazovkách a ako vyzerá.

4. Pre jednotlivé prvky navrhnúť správanie a naprogramovať logiku aplikácie.

5. Nahrať program do mobilného telefónu (aplikácia AI2 Companion) prípadne do simulácie v PC.

Nahranie programu

1. Cez menu: Connect – AI Companion.

2. Na mobilnom telefóne spustiť AI2 Companion.

3. Naskenovať čiarový kód / zadať kód ručne.

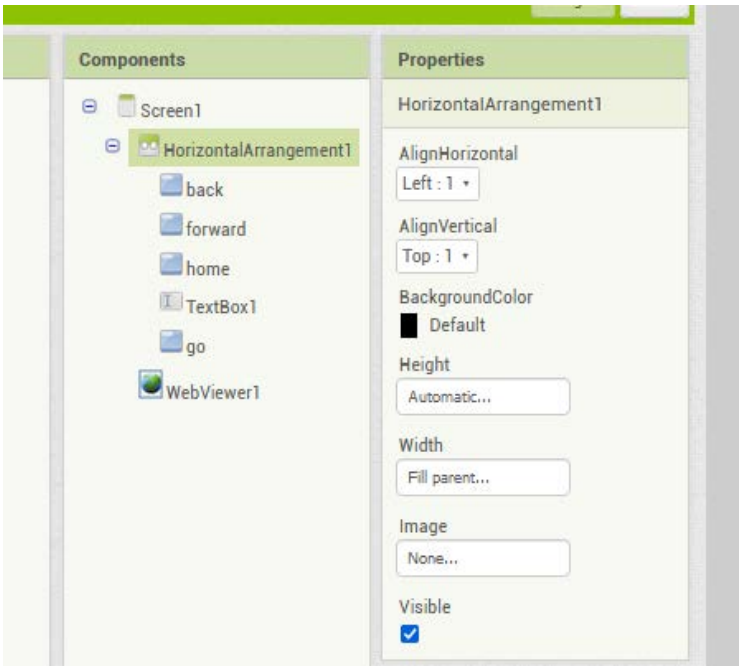
4. Spustenie aplikácie.

3.2 Príklad - internetový prehliadač v mobile

Cieľom je vytvoriť mobilnú aplikáciu, kde by sa využíval internetový prehliadač s možnosťami ako prechod na úvodnú obrazovku, dopredu, naspäť a prechod na zvolenú webovú adresu.

1. vložíme oblasť horizontálneho nastavenia pre tlačidlá

(Layout - HorizontalArrangement) ; Výška – Automatic ; Šírka – Fill parent (vyplní celý riadok)



2. vložíme objekty button do horizontálneho zarovnania

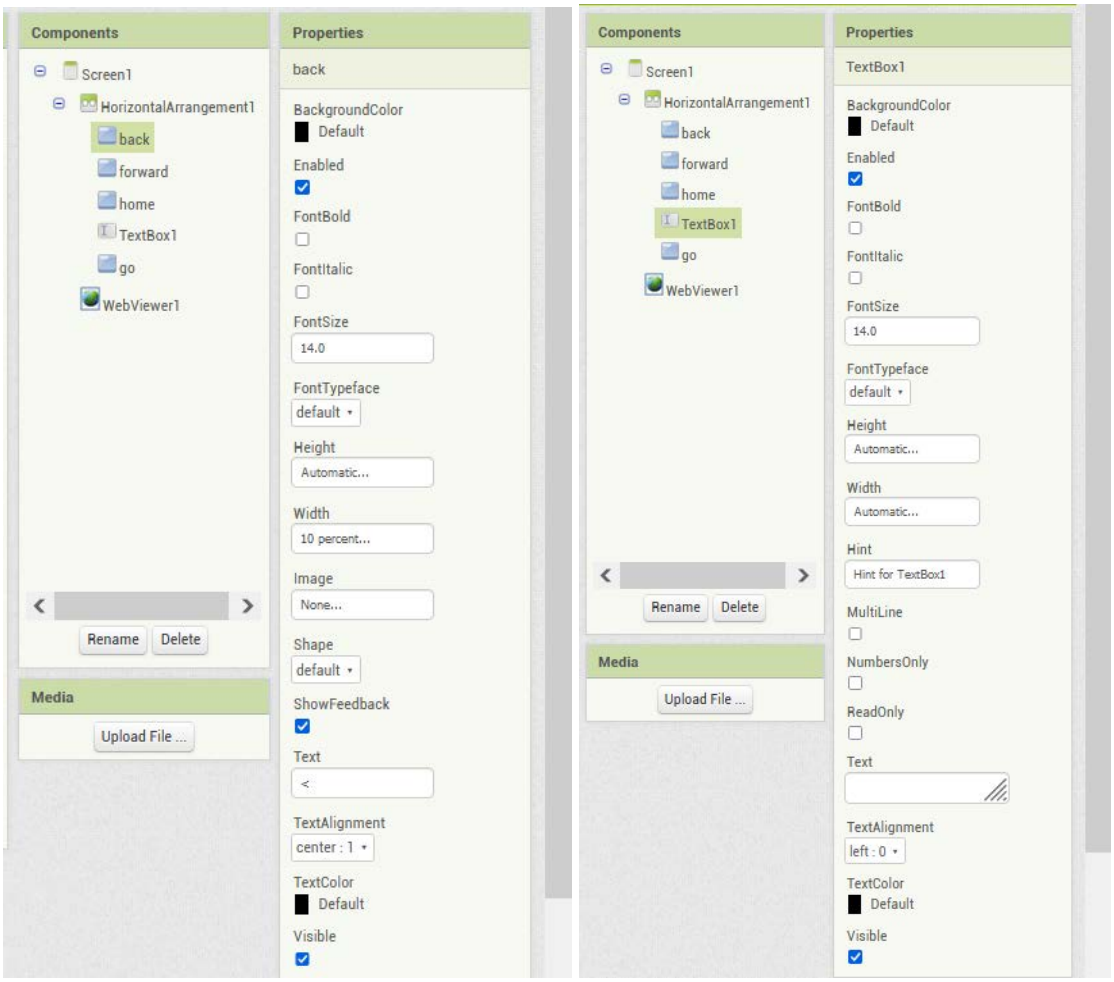
(User Interface - Button) ; pre všetky ovládacie tlačidlá a nastavíme im 10% Width a text pre zobrazenie na tlačidle (prípadne ďalšie vlastnosti – farba, písmo...)

3. vložíme textové pole pre zadanie webovej adresy

(User Interface - TextBox)

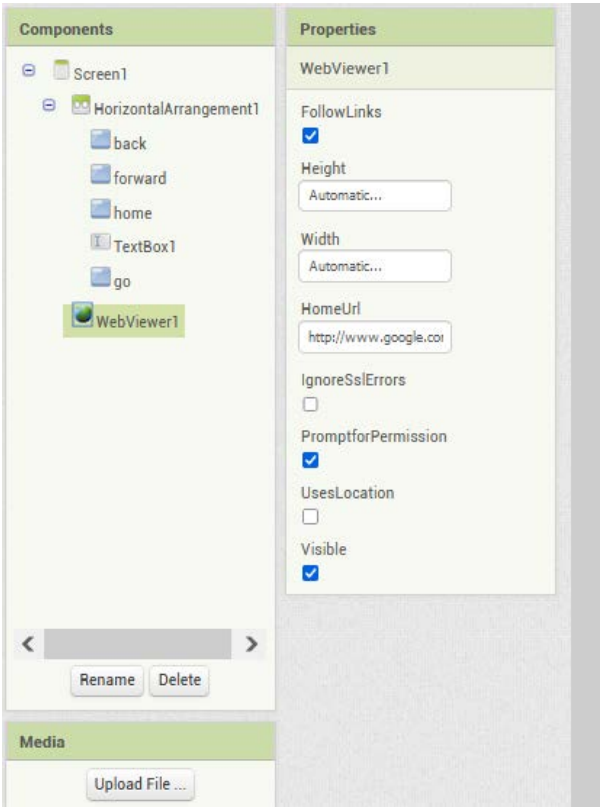
4. vložíme objekt internetového prehliadača

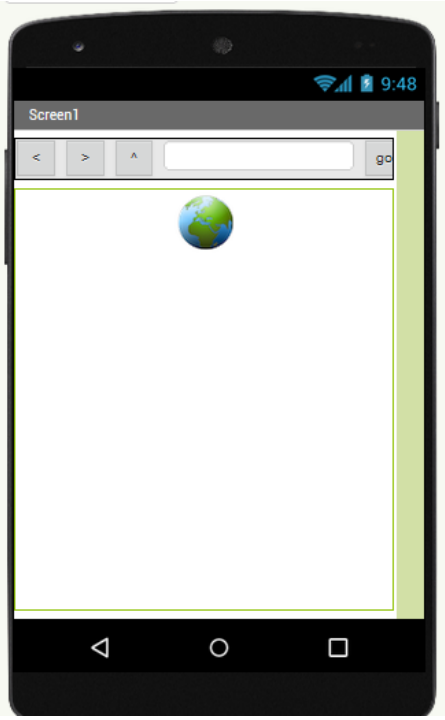
(User Interface - WebViewer) ; zadáme domovskú adresu (HomeUrl)



5. v oblasti použitých komponentov premenujeme jednotlivé tlačidlá podľa toho, na čo majú slúžiť.

6. prepneme do rozhrania Blocks



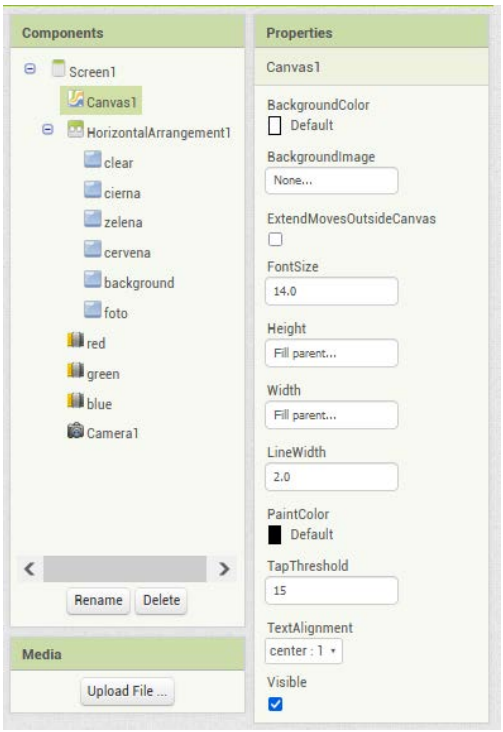


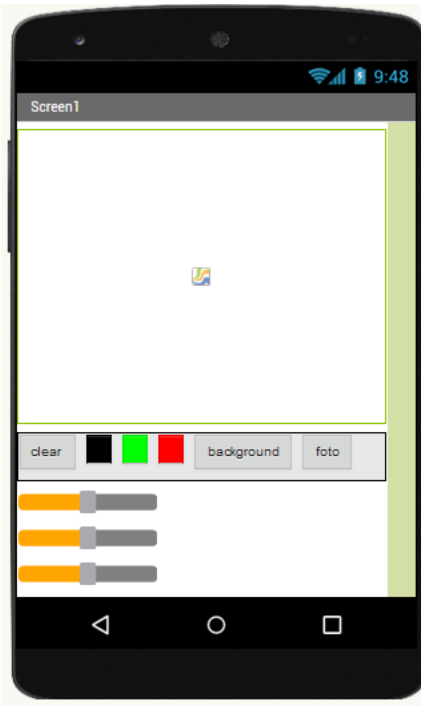
Pre jednotlivé tlačidlá priradím metódy po kliknutí podľa potreby – prechod na url adresu zadanú do TextBoxu, návrat na predchádzjúcu adresu, prechod na ďalšiu adresu, prechod na domovskú adresu

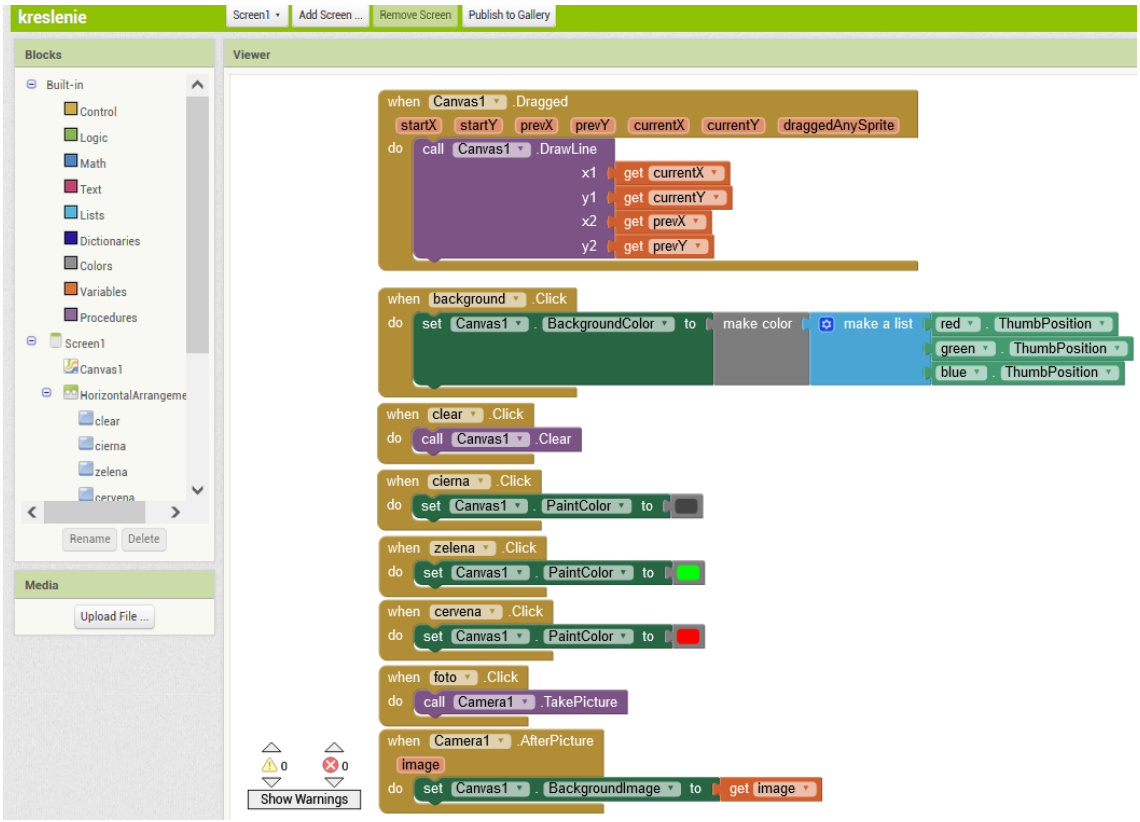
3.3 Príklad - kreslenie v mobile

Cieľom je vytvoriť mobilnú aplikáciu, kde by sme vytvorili kresliace plátno a mohli nastavovať farbu čiary z preddefinovaných farieb, ktorou budeme kresliť, farbu pozadia nastaviteľnú v režime RGB a použiť fotoaparát a následne snímku nastaviť ako pozadie.

- vložíme objekt Canvas zo skupiny Drawing and Animation a nastavíme vlastnosti
- vložíme tlačidlá pre mazanie, pozadie, fotoaparát a voľbu farieb pera
- vložíme 3 objekty Slider (User Interface) pre nastavenie farieb
- vložíme objekt Camera (skupina Media)
- v zobrazení Blocks vytvoríme program pre aplikáciu.
- farbu pera volím z farieb čierna, zelená, červená
- farbu pozadia miešam v režime RGB
- tlačidlo Clear maže pozadie
- tlačidlo foto použije fotoaparát a nastaví ako pozadie



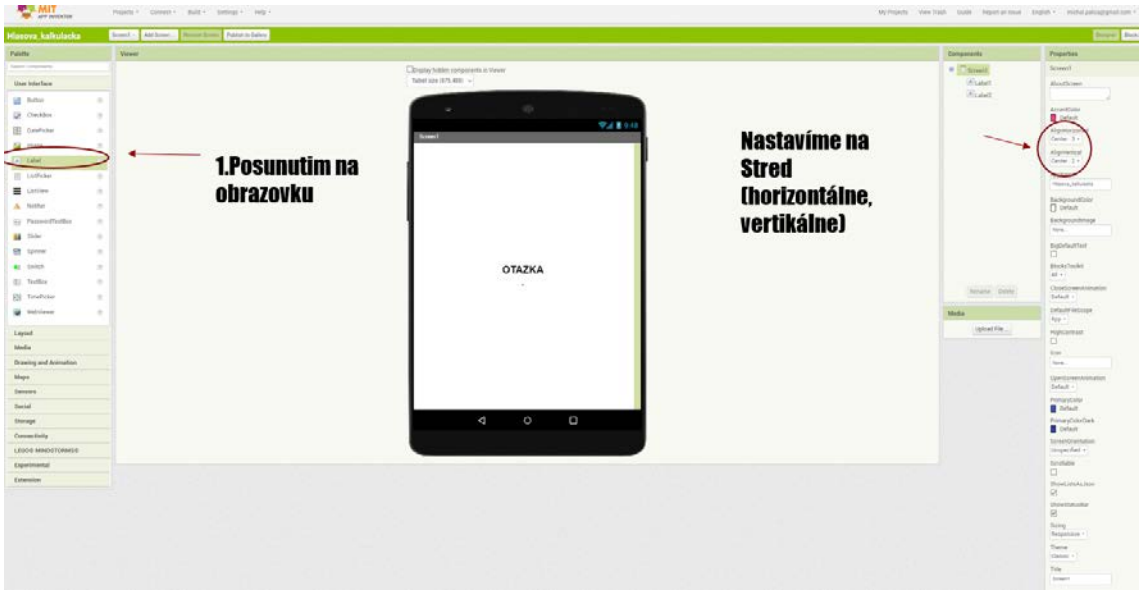




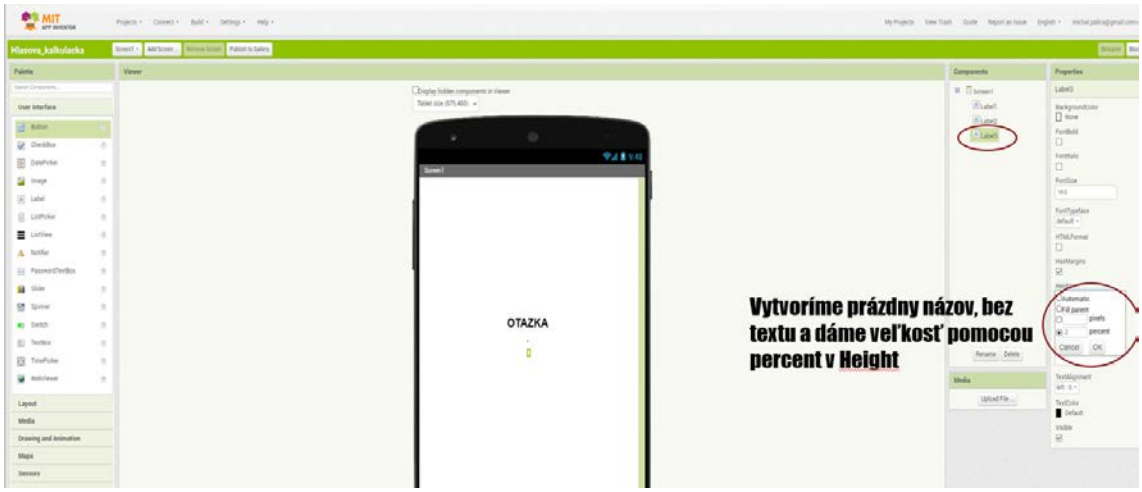
3.4 Hlasová kalkulačka

- Ide o jednoduchý program kde pomocou hlasu budeme vedieť používať základne matematické operácie ako sčítanie, odčítanie, delenie a násobenie
- Čísla ktoré chceme použiť na operáciu zadávame hlasom využitie google speech to text
- Naprogramovať komplexný program ktorý bude pomocou hlasu ovládať program alebo zariadenia
- Prepojenia z arduinom a rôznymi senzormi
- Komplexná aplikácia na ovládanie domáceho smart home(žiadna aplikácia 3. strany)

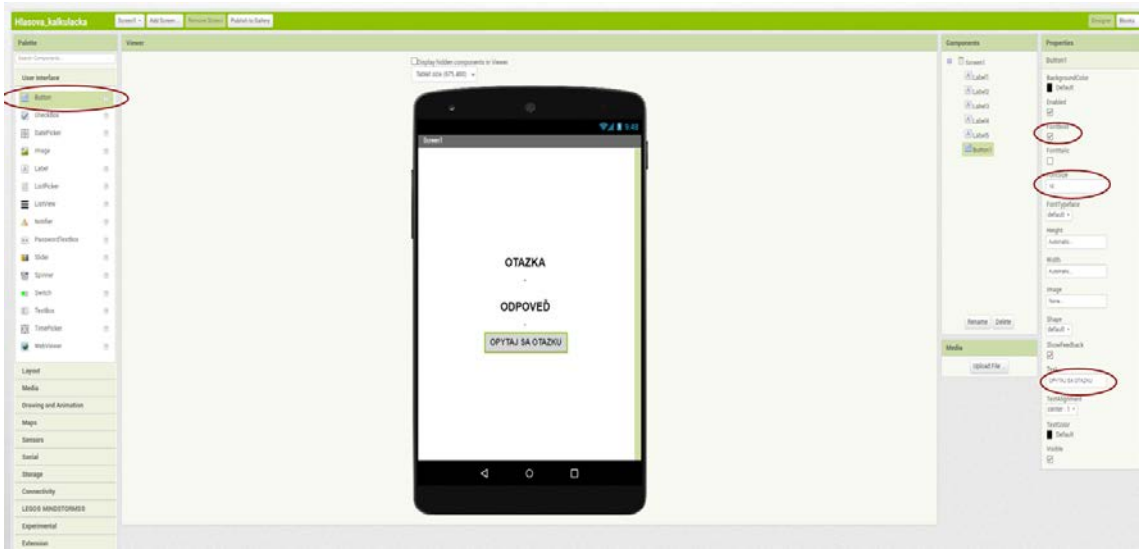
Grafický postup:


1.Posunutim na obrazovku
OTAZKA
Nastavíme na Stred (horizontálne, vertikálne)

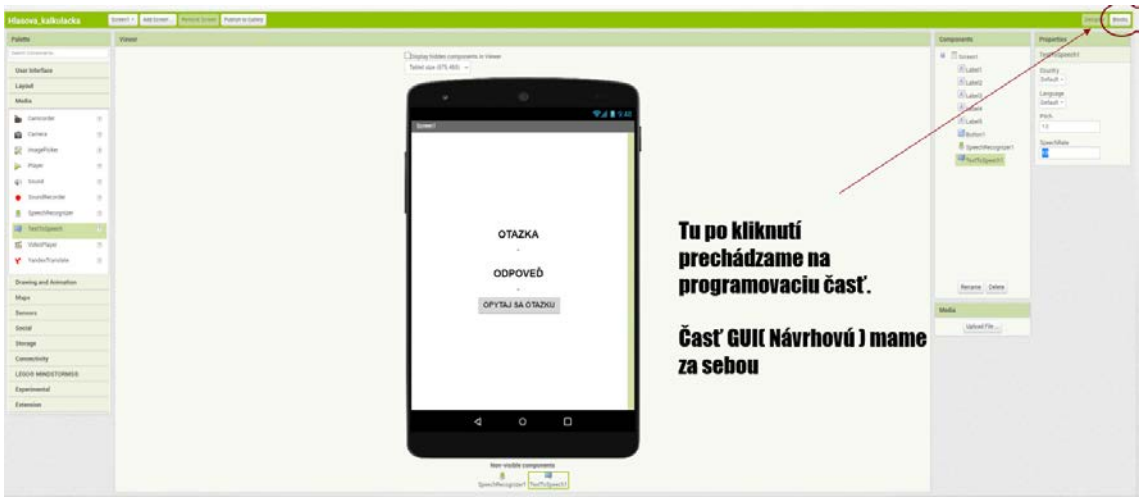

Prekliknutie na prvý názov(Label)
Tučne písmo
Veľkosť písma
OTAZKA
Nadpis čo tam chceme

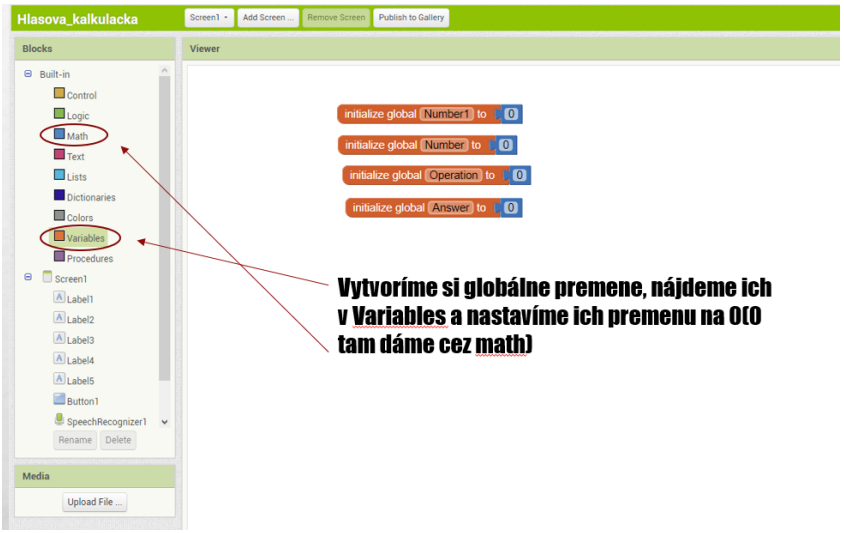

OTAZKA
Vytvoríme prázdny názov, bez textu a dáme veľkosť pomocou percent v Height


OTAZKA
ODPOVEĎ
Označíme label 1 a dáme ctrl+c a ctrl+v a vytvoríme nám identicky nastavený label ktorý posunieme na spod a do textu mu napíšeme do názvu *odpoveď*
To iste urobíme aj s pomlčkou


OTAZKA
ODPOVEĎ
OPYTAJ SA OTAZKU


Potiahnem na plochu obidve ľubovoľne na plochu)
OTAZKA
ODPOVEĎ
OPYTAJ SA OTAZKU
Tu ich vidíme že sú na ploche


OTAZKA
ODPOVEĎ
OPYTAJ SA OTAZKU
Tu po kliknutí prechádzame na programovaciu časť.
Časť GUI(Návrhovú) mame za sebou


Hlasova_kalkulacka
Screen1
Add Screen ...
Remove Screen
Publish to Gallery
Blocks
Viewer
Built-in
Control
Logic
Math
Text
Lists
Dictionaries
Colors
Variables
Procedures
Screen1
Label1
Label2
Label3
Label4
Label5
Button1
SpeechRecognizer1
Rename
Delete
Media
Upload File ...
initialize global Number1 to 0
initialize global Number to 0
initialize global Operation to 0
initialize global Answer to 0
Vytvoríme si globálne premene, nájdeme ich v Variables a nastavíme ich premenu na 0(0 tam dáme cez math)

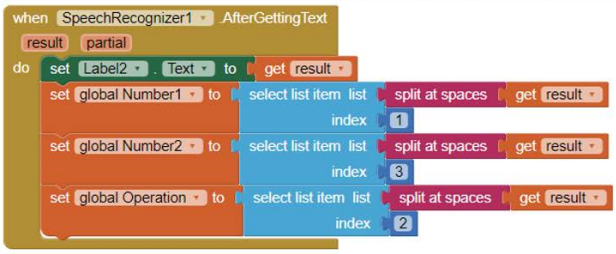

7 + 5 si musíme rozdeliť na 3 premene a preto sme dopísali ručne tieto parametre aby sme si vysvetlili ktoré sú to premene
OTAZKA
7 + 5
ODPOVEĎ
-
OPYTAJ SA OTAZKU
initialize global Number1 to 0
initialize global Number2 to 0
initialize global Operation to 0
initialize global Answer to 0
Čisto vymyslený príklad

1.

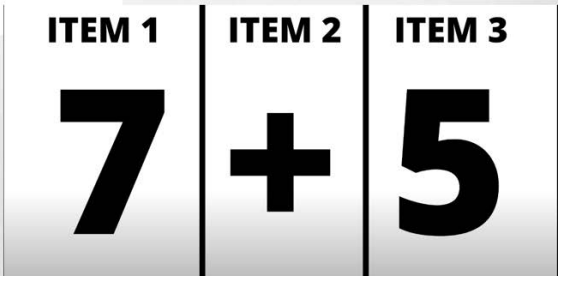
2.

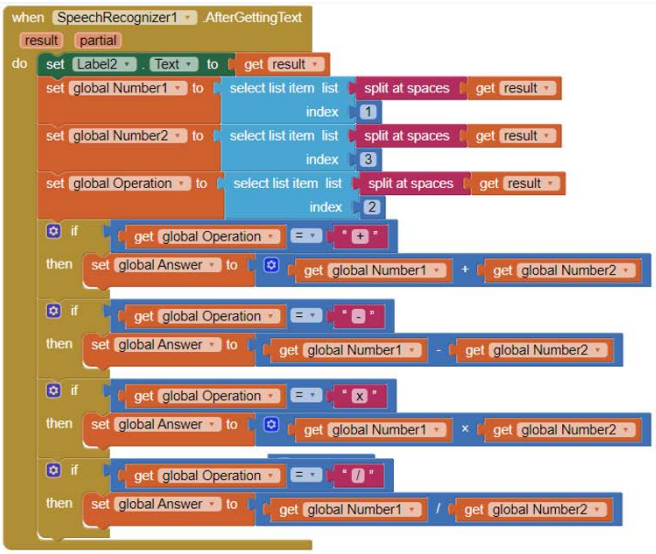
3.

Začíname vytvárať podmienky

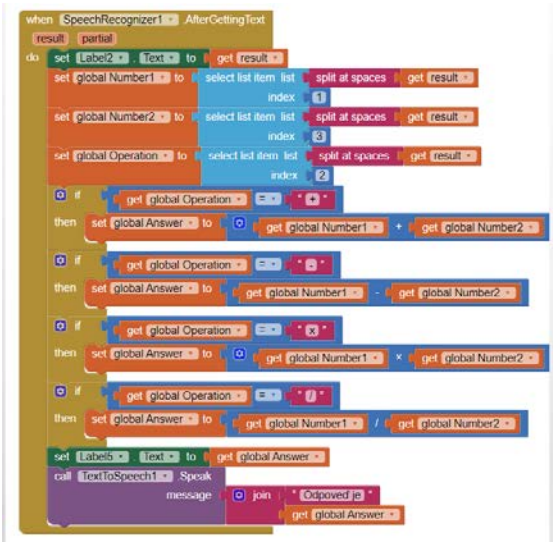


Vidíme pozície

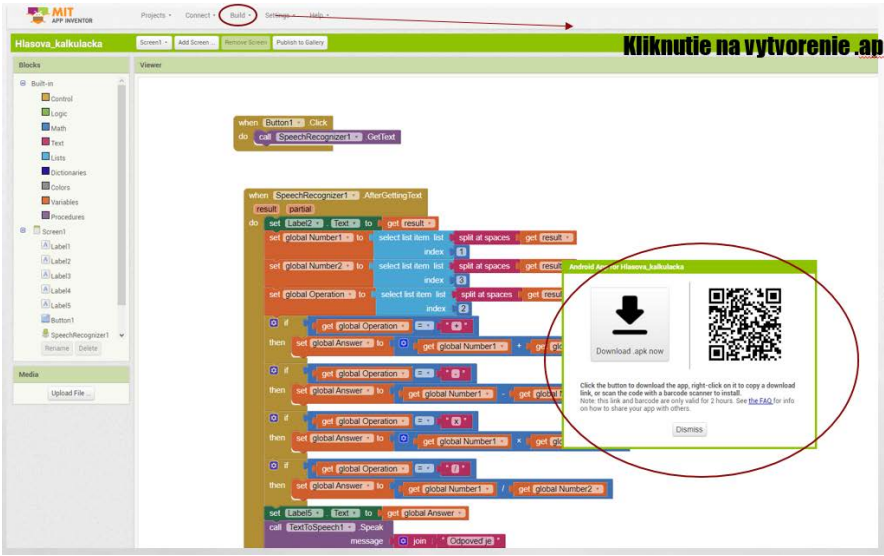




Vytvárame všetky podmienky ktoré sme si zadali na začiatku



Nakoniec zadáme čo ma robiť Label 5 kde ma byt výsledok



Kliknutie na vytvorenie .apk alebo .aab

Stiahnutie do počítača alebo cez QR kód do mobilu